UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Jankovec

Porazdeljen sistem za avtomatizacijo pametne hiše

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Aleš Jankovec

Porazdeljen sistem za avtomatizacijo pametne hiše

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: viš. pred. dr. Robert Rozman

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Porazdeljen sistem za avtomatizacijo pametne hiše.

Izdelajte zasnovo porazdeljenega sistema za avtomatizacijo pametne hiše. Sistem naj bo sestavljen iz centralnega strežnika, na katerega so povezane različne krmilne enote. Te naj opravljajo vlogo vmesnika med centralnim strežnikom in končnimi napravami v pametni hiši, kot so npr. luči, senčila, ogrevalne in prezračevalne naprave. Razvijte tudi ustrezno podatkovno bazo potrebno za delovanje sistema. Zagotovite možnost krmiljenja in komunikacije z vsemi napravami iz centralnega strežnika. Omogočite daljinsko krmiljenje sistema s spletno aplikacijo. Zasnovo sistema tudi dejansko implementirajte in pri tem zagotovite čim splošnejšo uporabnost programskih modulov in celotnega sistema na najrazličnejših računalniških platformah.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani Aleš Jankovec, z vpisno številko **63040219**, sem avtor diplomskega dela z naslovom:

Porazdeljen sistem za avtomatizacijo pametne hiše

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Roberta Rozmana,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 23. september 2014 Podpis avtorja:

Zahvala

Rad bi se zahvalil svojemu mentorju, viš. pred. dr. Robertu Rozmanu, za pripravljenost prevzeti mentorstvo, strokovno vodenje in pomoč pri izdelavi diplomskega dela.

Zahvaljujem se tudi svoji partnerki in svojim staršem za vso podporo in pomoč.

Hvala tudi vsem prijateljem in sodelavcem, ki so mi na kakršenkoli način pomagali pri pripravi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
JVM	java virtual machine	virtualno okolje za izvajanje java kode
IDE	integrated development environment	integrirano razvojno okolje
JSON	javascript object notation	format za prenos programskih razredov v serializirani obliki
JSON-RPC	remote procedure call protocol	protokol za oddaljen klic postopkov
SQL	structured query language	strukturirani povpraševalni jezik za delo s podatkovnimi bazami
TCP	transmission control protocol	protokol za nadzor prenosa
IP	internet protocol	internetni protokol
DL	database layer	podatkovni nivo
DAL	database access layer	nivo podatkovnega dostopa
BLL	business logic layer	nivo poslovne logike
SOAP	simple object access protocol	protokol za izmenjavo podatkov
GUI	graphical user interface	grafični uporabniški vmesnik
JAR	java archive	tip arhiva z java razredi
CRON	time-based job scheduler	časovnik za izvedbo nalog
ASCX	ASP.NET Web User Control	ASP.NET spletna uporabniška kontrola

Povzetek

Cilj diplomske naloge je zasnovati in implementirati porazdeljen, modularen, dinamičen in hkrati splošen sistem pametne hiše. Rezultat dela je porazdeljen sistem, ki ga sestavljajo centralni strežnik v povezavi s krmilnimi enotami. Te upravljajo delovanje končnih naprav (luč, žaluzije) in jih s tem vključujejo v sistem. Njegov pomemben del so tudi naprave za oddaljeno upravljanje sistema preko spletnega vmesnika. Razvit in implementiran je podatkovni model in ustrezna podatkovna baza. Centralni strežnik in spletni vmesnik sta implementirana z n-tirno arhitekturo, ki omogoča modularno zgradbo. Programski modul krmilne enote je sestavljen iz dveh delov: strežnika in vhodno/izhodne dinamične knjižnice. Strežnik krmilne enote je neodvisen od platforme in ga lahko uporabimo na vseh tipih krmilnih naprav, medtem ko dinamična knjižnica vsebuje posebnosti vhodno/izhodnih priključkov krmilne enote in je od nje odvisna.

Ključne besede: pametna hiša, porazdeljena zasnova, centralna enota, krmilna enota, strežnik

Abstract

The goal of the diploma thesis is to design and implement a distributed, modular, dynamic and at the same time general smart-house system. The result of the work is a distributed system composed of a central server in conjunction with the control units. These manage the operation of the end-devices (lights, blinds) and connect them to the system. An important part are devices for remote system management via web interface. A data model and the corresponding database are also developed and implemented . The central server and the Web interface are implemented with multi-tier architecture, which allows a modular structure. The control unit software module is composed of two parts: the server and the input/output dynamic library. The server control unit is independent from the platform and can be used on all types of control devices. The dynamic library contains the specifics of the input/output terminals of the control unit and is dependent on it.

Keywords: smart home, distributed design, central unit, control unit, server

Poglavje 1 Uvod

Pametna hiša ni samo hiša, ki jo upravljaš iz naslonjača preko telefona ali tabličnega računalnika, ampak tudi avtomatizirano izvajanje različnih opravil. To pomeni, da dobro zasnovan sistem samodejno izvaja opravila, ki omogočajo prihranek energije, povečajo udobje in poskrbijo za varnost brez posredovanja uporabnika.

Danes je na domačem trgu že nekaj ponudnikov sistemov za pametno upravljanje hiše, še več izbire ponuja tujina. Naročnik mora pri izbiri takega sistema predvsem dobro preučiti svoje potrebe, saj višji kot je nivo avtomatizacije v hiši, bolj kompleksen je sistem in s tem večja možnost okvare ter posledično zahtevnost vzdrževanja. Pri iskanju primernega sistema za svojo hišo smo iskali sisteme, ki so čimbolj odprti, dinamični, predvsem pa čimbolj modularni in imajo možnost povezave več različnih sistemov. Izkazalo se je, da je večina obstoječih sistemov preveč zaprtih.

Tako smo kaj hitro prišli do ideje o razvoju in postavitvi lastne rešitve za avtomatizacijo hiše, ki bo omogočala nadzor in upravljanje vseh naprav znotraj hiše v centralnem nadzornem sistemu.

V začetku bomo na kratko predstavili ključna področja pametne hiše. Pred predstavitvijo implementacije programskih modulov sistema, pa je predvsem dobro razumeti pomen posameznih elementov v sistemu. Pomembno je poznavanje podrobnosti naprav, ki jih priključujemo v sistem. V zadnjem delu diplomskega dela si bomo pogledali praktično implementacijo takega sistema.

Namen diplomskega dela je predstavitev zasnove in izdelave porazdeljenega sistema za pametno upravljanje hiše. Cilj diplomskega dela je na podlagi zasnovane porazdeljenega sistema pametne hiše izdelati konkreten primer sistema.

Poglavje 2 Področja avtomatizacije pametne hiše

Preden se lotimo razmišljanja o zasnovi sistema, je pomembno spoznati, kaj taki sistemi doprinesejo pri udobju in uporabi hiše. Poznavanje področij, ki jih obravnavajo taki in podobni sistemi za upravljanje hiše, je zelo pomembno pri arhitekturni zasnovi sistema, saj vsako področje vsebuje specifične naprave s specifičnimi nalogami. V tem poglavju si bomo ogledali ključna področja in naprave, ki sestavljajo tipičen sistem pametne hiše.

2.1 Razsvetljava

Avtomatizacija razsvetljave v pametni hiši je zelo pomembno področje pametnih sistemov. Krmiljenje pametne razsvetljave ne pomeni, da lahko preko računalnika, pametnega telefona ali tabličnega računalnika samo nadzorujemo stanje naprave, jo prižigamo ali ugašamo. Pri tem je pomembna predvsem avtomatizacija opravil.

Sistem lahko uporabljamo na naslednje načine, in sicer za:

- nadzorovanje stanja luči,
- oddaljeno krmiljenje luči,
- izklapljanje ali vklapljanje vseh luči v sistemu z enim samim klikom,
- avtomatizacijo opravil.

Krmiljenje razsvetljave se odlično obnese v kombinaciji z raznimi tipali, pri čemer so predvsem ključna svetlobna tipala in tipala gibanja.

2.1.1 Primer prižiganja vrtne luči

Za avtomatizacijo prižiganja vrtne luči mora biti sistem pametne hiše zastavljen tako, da omogoča izvajanje določenih opravil glede na stanje tipal. V našem primeru bi bilo opravilo prižigati/ugašati luči glede na vrednost svetlobnega tipala. Opravilo mora biti v sistemu določeno tako, da se luči samodejno prižgejo, ko tipalo svetlobe zazna padec osvetljenosti pod

določeno mejo (Slika 2.1). V nasprotnem primeru (tipalo svetlobe zazna vrednost osvetljenosti nad določeno mejo) pa se luči ugasnejo.

Slika 2.1: Prikaz prižiganja vrtnih luči [1].

2.1.2 Primer prižiganja luči v kombinaciji s tipalom gibanja

Imamo svetlobno tipalo, ki meri osvetljenost prostora, in tipalo gibanja, ki zaznava premike v prostoru. Če svetlobno tipalo zaznava osvetljenost pod določeno mejo, se luči prižgejo le v primeru, ko tipalo gibanja zazna premik. Če pa svetlobno tipalo zaznava, da je prostor dovolj svetel, se luči ne prižgejo.

Slika 2.2: Prižiganje luči v temnem prostoru ob gibanju [2].

Takšnih in podobnih primerov uporabe avtomatike v razsvetljavi je še veliko.

2.2 Ogrevanje in klimatizacija

Ogrevanje in klimatizacija prostorov je prav tako področje, kjer lahko pametna hiša pripomore k zmanjšanju porabe energije. To lahko dosežemo predvsem z merjenjem notranje in zunanje temperature in s pravilnim krmiljenjem centralnega sistema za ogrevanje ali klimatizacije. Z merjenjem temperature vseh prostorov v hiši lahko ugotavljamo, kateri prostor ima največje energetske izgube. Za merjenje temperature uporabljamo notranje in zunanje termometre, ki morajo biti del sistema.

Prav tako lahko poleg notranje in zunanje temperature nadzorujemo temperaturo centralne peči, temperaturo vode v grelniku sanitarne vode … Prižigamo lahko centralno peč, črpalke za ogrevanje ...

2.3 Multimedija

Multimedija je področje, ki je pri pametni hiši specifično predvsem z vidika zahtevnosti krmiljenja in nadzora. Multimedijske naprave morajo biti prilagojene za posamičen pametni sistem. Krmilimo lahko radio, CD-predvajalnike, TV-predvajalnike…

Pri tem področju se sistem pametne hiše uporablja za pripravo prostora za določen namen: med gledanjem filmov se npr. spustijo žaluzije, ugasnejo luči in radio. Poslušanje glasbe: prižge se nežna glasba, zatemnijo se luči, spustijo se žaluzije …

2.4 Varnost doma

Na področju varnosti se sistem pametne hiše uporablja predvsem pri vklapljanju alarma in ugašanju luči v primeru odsotnosti. Lahko ga uporabljamo tudi za obveščanje o odprtih oknih ali vratih v primeru odhoda iz hiše. Pametna hiša se uporablja tudi pri obveščanju preko sms-sporočil ali elektronske pošte v primeru sprožitve alarma ali zaznave gibanja. V primeru daljše odsotnosti bi lahko imeli tudi opravila, ki bi samodejno prižigala luči in s tem odganjala morebitne vlomilce.

2.5 Senčila

Senčila so pomembna pri vzdrževanju toplotnega udobja nizkoenergijskih ali pasivnih hiš. Napredni sistemi poleg merjenja zunanje in notranje temperature upoštevajo tudi stanje sonca. Senčila poleti preprečujejo pregrevanje hiše, pozimi pa lahko pripomorejo pri zmanjšanju izgub preko oken. Seveda morajo biti naprave med seboj dobro usklajene, za kar mora poskrbeti nadzorni sistem.

2.5.1 Primer krmiljenja senčil

Na kratko bomo pogledali, kako bi lahko pametna hiša krmilila senčila v zimskem času.

V sistemu imamo zunanje temperaturno tipalo in tipalo osvetljenosti. V primeru, da je temperatura zelo nizka in osvetljenost pod ali pa nad določeno mejo, se senčila spustijo ali dvignejo. Tako preprečijo nepotrebne toplotne izgube.

Slika 2.3: Primer odpiranja senčil ob svetlobi [3].

2.6 Ostala področja

Ostala področja, ki jih zajema pametna hiša, so:

- krmiljenje prezračevanja,
- nadzor raznih tipal, vremenskih postaj,
- uravnavanje vlažnosti shramb ...

Področij uporabe pametnih hiš je še veliko, predvsem pa mora biti sistem nastavljen tako, da so naprave in področja med seboj čimbolj ustrezno koordinirane.

Poglavje 3 Zasnova porazdeljenega sistema pametne hiše

Porazdeljen sistem pametne hiše sestavljata vsaj dva ali več aktivnih modulov - računalnikov oziroma vgrajenih sistemov na različnih lokacijah. Moduli v porazdeljenem sistemu imajo svoj lokalni pomnilnik in medsebojno komunicirajo z uporabo različnih prenosnih poti. V našem primeru je to lokalno domače omrežje. V nadaljevanju si bomo pogledali vse tipe naprav oziroma računalnikov, ki jih potrebujemo v porazdeljenem sistemu pametne hiše.

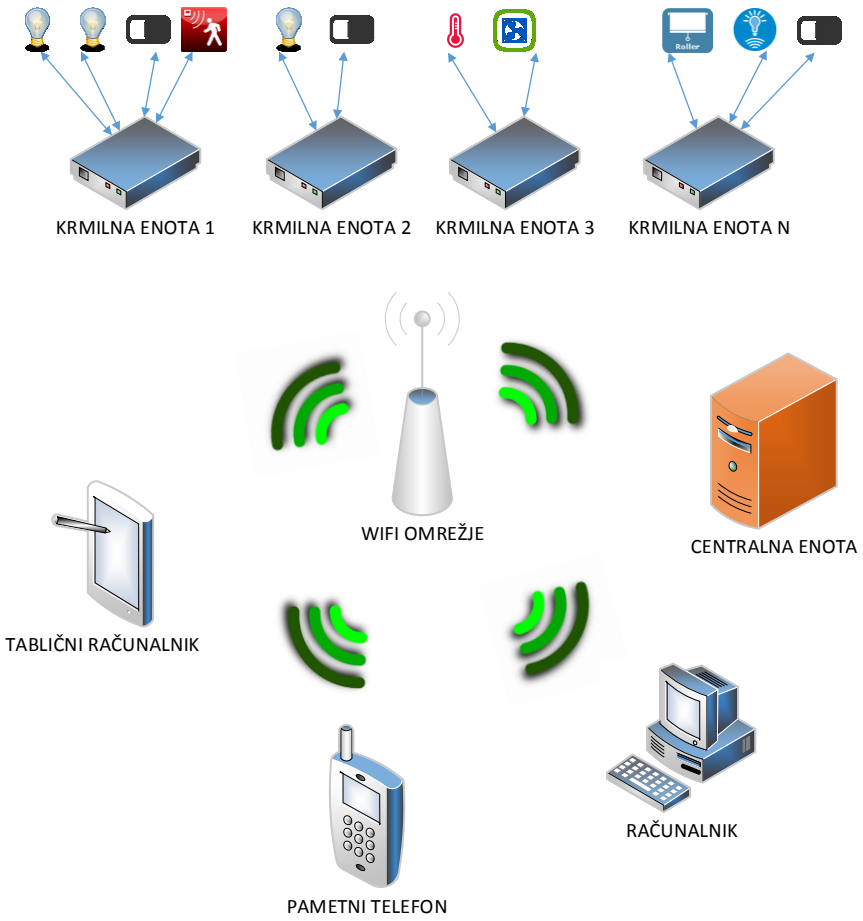
3.1 Naprave v porazdeljenem sistemu

Arhitekturno je rešitev zasnovana z naslednjimi sestavnimi deli:

- s centralno enoto – temu bi lahko rekli tudi centralni strežnik,
- s krmilnimi enotami – računalniki v vlogi krmilne enote,
- z napravami za upravljanje in nadzorovanje sistema: tablični računalnik, pametni telefon, prenosni računalnik …
- s končnimi napravami, ki so priključene v sistem: luči, stikala, tipala, motorji …,
- s pripadajočo komunikacijsko opremo za povezavo naprav.

Arhitektura sistema je prikazana na sliki 3.1. Prednost tako porazdeljenega sistema je, da ni potrebno končnih naprav povezovati v eno točko hiše. Tako se znebimo velikega števila odvečnih napeljav. Po celotni hiši lahko zato postavimo poljubno veliko krmilnih enot, na katere so priključene dejanske končne naprave.

Kot nam prikazuje slika 3.1, imamo v sistemu različne tipe naprav, ki so med seboj lahko povezane v brezžično lokalno omrežje. Dobrodošla lastnost brezžičnega omrežja je povezava naprav brez uporabe vodnikov, kar je koristno pri sistemu z veliko napravami. Če bi imeli v vsakem prostoru svojo krmilno enoto ali več le-teh, se z brezžičnim omrežjem znebimo veliko odvečnih napeljav žičnega omrežja.



Slika 3.1: Naprave sistema pametne hiše.

3.1.1 Centralna enota

Centralna enota predstavlja možgane sistema. Sestavljena je lahko iz enega ali več dejanskih računalnikov, ki med seboj komunicirajo. Če uporabimo samo en računalnik, mora odigrati vlogo aplikativnega, spletnega in podatkovnega strežnika. Če pa centralni računalnik tvori več računalnikov, lahko vsak računalnik odigra ločeno vlogo. Zagotoviti moramo samo, da je na jih nameščen operacijski sistem Windows server. V lokalno omrežje pa je lahko priključen žično ali pa brezžično. V celotnem sistemu moramo imeti samo eno centralno enoto, ki mora biti aktivna brez prekinitev.

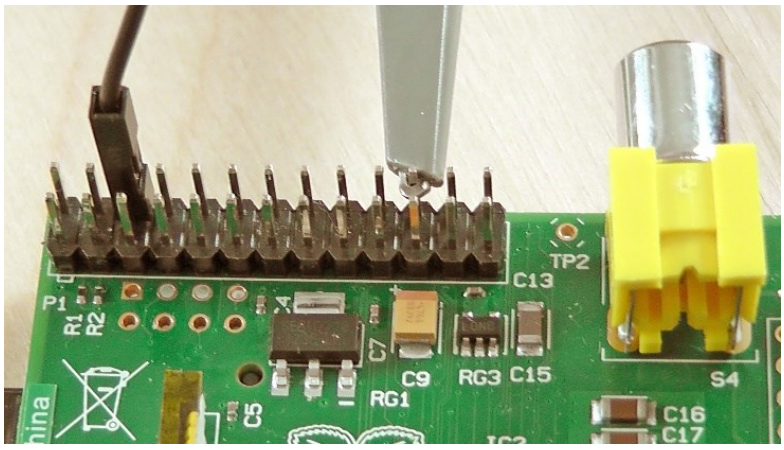
Naloga centralne enote je, da s pomočjo različnih programskih sklopov, ki so nameščeni na operacijski sistem, naredi naslednje:

- hrani podatke o celotnem sistemu, napravah, krmilnih enotah …,
- povezuje celotni sistem,

- procesira zahteve aplikacij, ki se povezujejo nanjo,
- se povezuje na posamezne krmilne enote,
- pridobiva podatke iz teh enot in jih obdeluje.

3.1.2 Krmilna enota

Krmilna enota je naprava, preko katere je mogoče v sistem priključiti končne naprave. Njena osnovna naloga je, da je sposobna s programsko opremo upravljati končno napravo in komunicirati s centralnim strežnikom. Za priključitev končne naprave potrebuje enota vhodno/izhodne priključke - Slika 3.2.



Slika 3.2: Vhodno/izhodni priključki krmilne enote [4].

Za krmilno enoto lahko uporabimo katero koli napravo, ki izpolnjuje naslednje zahteve:

- imeti mora zadostno število vhodno/izhodnih priključkov,
- omogočati mora brezžično ali žično povezavo v domače omrežje,
- omogočati mora namestitev programskega paketa Java virtual machine (JVM),
- enota mora biti čim manjša.

Poglejmo si nekaj najbolj znanih kartičnih računalnikov, ki ustrezajo zgornjim zahtevam in so primerni za uporabo v opisanem sistemu pametne hiše:

- Raspberry Pi model A, B ali B+,

- Beagle Bone Black,
- Arduino Yun …

Poleg naštetih kartičnih računalnikov lahko za krmilnik uporabimo kar PC-računalnik, ki pa je bolj primeren za upravljanje programov kot končnih naprav. Lahko pa ga s kombinacijo pravilnega krmilnika, ki ponuja vhodno/izhodne priključke, uporabimo tudi za krmiljenje končnih naprav, vendar se pri tem pojavi težava velikosti in tudi cene.

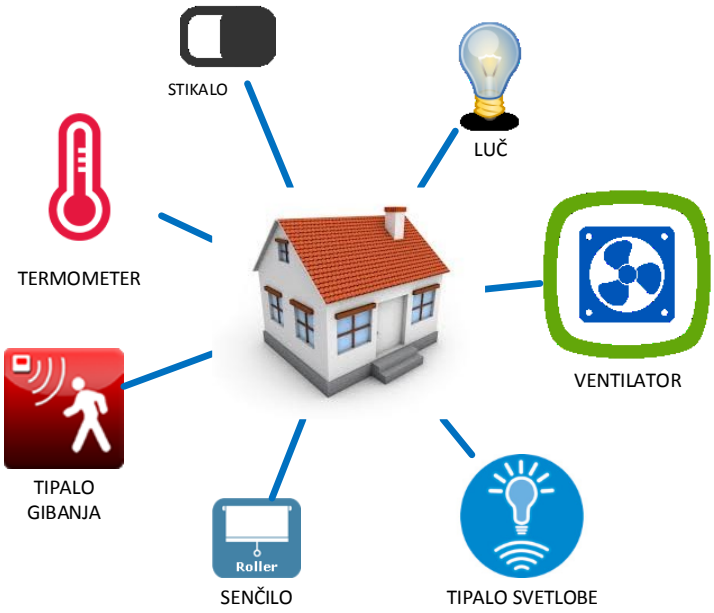
Sistem, kot smo ga zasnovali, lahko vključuje poljubno število krmilnih enot.

3.1.3 Naprave za oddaljeno upravljanje

To so naprave, preko katerih končni uporabnik s pomočjo posebne aplikacije nadzoruje in upravlja sistem: predvsem so to pametni telefoni, tablični računalniki, prenosni in tudi klasični računalniki. Zasnovan sistem uporablja za upravljanje in nadzorovanje tudi spletno aplikacijo, zato lahko uporabimo katerokoli napravo, ki vsebuje sodobnejši spletni brskalnik. S tem je neodvisen od tipa naprave, ki jo bo uporabnik uporabil. Naprave lahko preko centralne enote krmilimo iz domačega brezžičnega ali žičnega omrežja ter oddaljenih lokacij, kar pripomore k splošni uporabnosti takega sistema.

3.1.4 Končne naprave

Končne naprave (Slika 3.3) so naprave, ki so v naš sistem priključene preko enega ali več vhodno/izhodnih priključkov krmilne enote. Te naprave so tudi del našega sistema. Mednje sodijo predvsem luči, tipala, senčila, alarmne naprave, okna, vrata …



Slika 3.3: Končne naprave v pametni hiši.

Kako take naprave priključimo v opisani sistem, si bomo ogledali v poglavju 3.3.

3.2 Arhitekturna zasnova porazdeljenega sistema

Postavitev pravilne arhitekture takega porazdeljenega sistema je ključnega pomena pri uporabnosti, modularnosti, fleksibilnosti in, seveda, dinamičnosti sistema. Na sliki 3.4 je prikazana arhitekturna zasnova sistema pametne hiše.

Sistem sestavljajo:

- centralna enota,
- večje število krmilnih enot,
- večje število končnih naprav,
- aplikacije oziroma naprave za upravljanje sistema.

V sistemu je samo ena centralna enota, krmilnih enot pa lahko imamo poljubno veliko. Vse končne naprave so v sistem priključene preko vhodno/izhodnih priključkov krmilne enote. Sistem lahko krmilimo preko spletne aplikacije, ali pa preko namenskih aplikacij, razvitih za specifičen tip naprave.

Na shemi (Slika 3.4) so vidni tudi vsi programski moduli, ki se nahajajo v posameznih sestavnih delih sistema:

- KRMILNA ENOTA

Programski modul krmilne enote smo arhitekturno razdelili na dva dela: prvi del je strežnik »dcServer«, drugi pa vhodno/izhodna dinamična knjižnica »ioLib«. Strežnik je tako povsem neodvisen od krmilne enote. Tako lahko isti strežnik poganjamo na kateremkoli tipu krmilne enote. Dinamična knjižnica »dcServer«-strežnika mora biti implementirana za vsak tip krmilne enote, saj ima vsak tip drugačne specifikacije vhodno/izhodnih priključkov. Naloge »dcServer«-modula pa so, da preko dinamične knjižnice krmili vhodno/izhodne priključke naprave. Bodisi prebira vrednosti in jih posreduje centralnemu strežniku bodisi nastavlja vrednosti, ki jih dobi od centralnega strežnika. »DcServer« izpolnjuje enostavne zahteve, ki jih dobi od centralnega strežnika, ne glede na vrsto naprave, ki je priključena na kateri vhodno/izhodni priključek.

- CENTRALNA ENOTA

Programska oprema centralnega strežnika je razdeljena na več modulov:

  - centralni strežnik »coreServer«,
  - spletno aplikacijo,
  - spletne storitve in
  - podatkovno bazo.

»CoreServer« je centralni strežnik, ki se mora »zavedati« vseh krmilnih enot, ki so v sistem registrirane, saj z njimi komunicira. Seveda se mora »zavedati« tudi vseh naprav, ki so priključene na krmilne enote. Obdelovati mora vse podatke, ki jih preko krmilne enote pridobi od naprave. Za hranjenje vseh informacij o krmilnih enotah in končnih napravah uporablja podatkovno bazo.

Spletna aplikacija je v osnovi namenjena za upravljanje in nadzorovanje sistema ter naprav, ki so priključene v sistem, namenjena pa je tudi administriranju sistema ter pregledovanju in urejanju uporabnikov sistema.

Spletne storitve so namenjene komunikaciji med strežnikom centralne enote in aplikacijami na napravah za upravljanje in nadzorovanje sistema. Preko spletnih storitev se prenašajo celotni razredi v serializirani obliki.

- NAPRAVE ZA UPRAVLJANJE SISTEMA

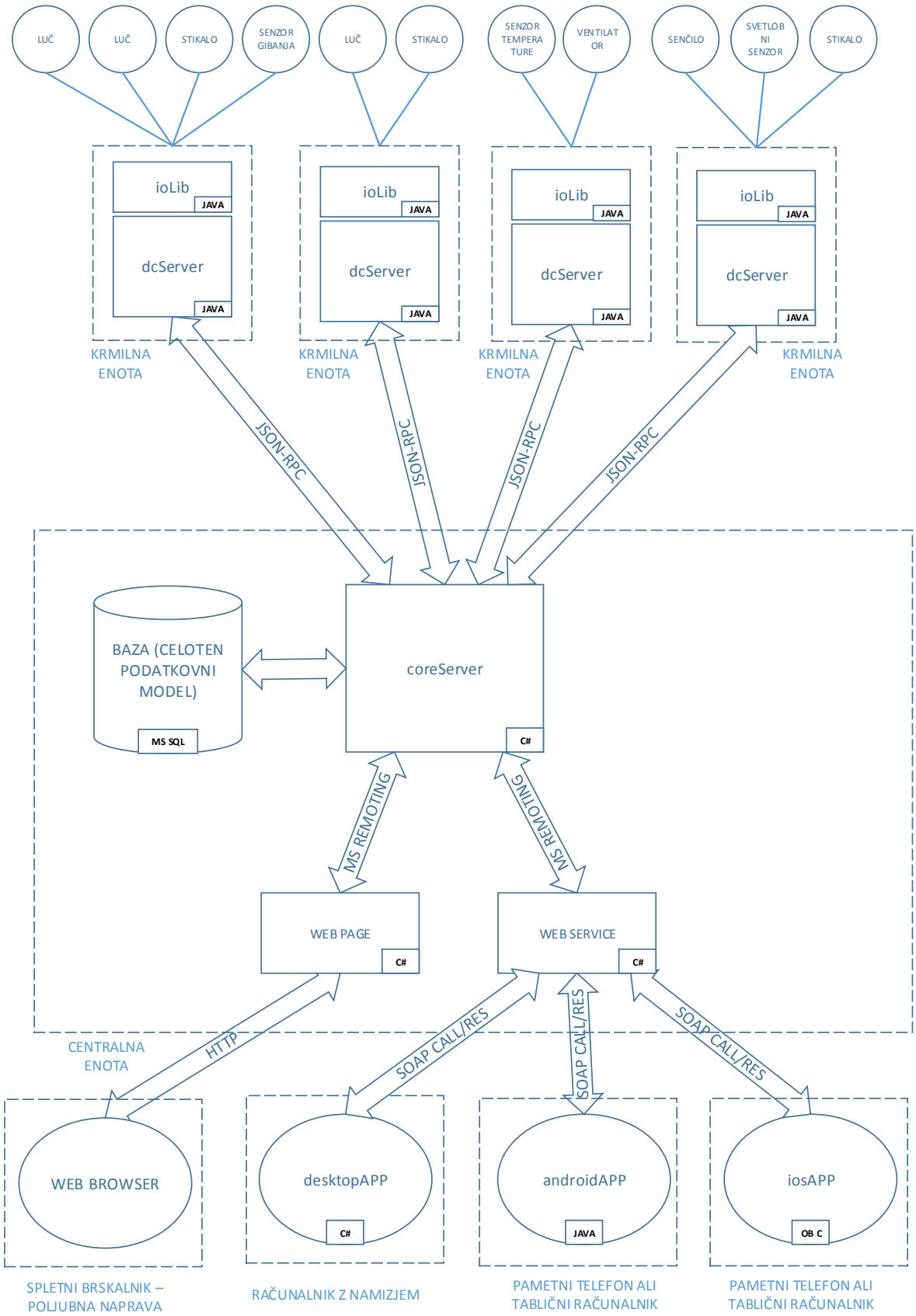
Programski moduli na posameznih aplikacijah, ki komunicirajo preko spletnih storitev (Slika 3.4), so grafični vmesniki do uporabnika, kjer lahko nadzoruje stanja posameznih naprav, sistema, predvsem pa lahko krmili naprave, ki so priključene v sistem.

Komunikacijski protokoli med programskimi moduli morajo temeljiti na serializaciji razredov. Za komunikacijo med centralnim strežnikom in krmilno enoto so predvideni klici oddaljenih postopkov, ki temeljijo na Json-formatu serializirane oblike razredov. Temu protokolu rečemo JSON-RPC. Prednost protokola je, da je enostaven in omogoča prenos serializiranih oblik razredov.

Za komunikacijo med programskimi moduli naprav za nadzor in upravljanje sistema ter centralnim strežnikom smo uporabili standardiziran protokol SOAP.

S programskega vidika so prednosti tako porazdeljenega sistema naslednje:

- v sistemu imamo samo en centralni strežnik, sistem je centraliziran;
- centralni strežnik ne komunicira neposredno s končnimi napravami, ampak samo z njim ustreznimi krmilnimi enotami;
- programska oprema krmilne enote se razvije enkrat in se lahko uporabi na poljubnem številu krmilnih enot;
- vsi podatki o stanjih, napravah so na enem mestu in dostop do njih ima samo centralni strežnik;
- s tako zasnovo sistema lahko v centralni enoti določimo tudi virtualno napravo, ki zajema poljubne naprave tudi iz različnih krmilnih enot, ki so fizično ločene. Tak primer bi bilo tipalo temperature, nameščeno na povsem drugem koncu hiše kot senčilo, ki bi jo krmilili z uporabo tega tipala;
- opravila se izvajajo na enem mestu - v centralnem strežniku.



Slika 3.4: Arhitektura porazdeljenega sistema pametne hiše.

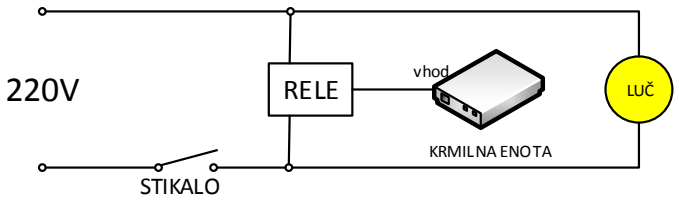
3.3 Vključitev končnih naprav v sistem

V tem poglavju bom predstavil, kako končno napravo priključiti na krmilno enoto in s tem povezati v sistem.

Vhodno/izhodni priključki krmilne enote so lahko analogni ali pa digitalni. Pri digitalnih priključkih imamo samo dve vrednosti: 1 ali pa 0. Pri analognih pa katero koli vrednost med 0 in zgornjo mejo, ki je odvisna od analogno digitalnega pretvornika. Ta meja je običajno 1023. Podatki so za krmilno enoto Raspberry Pi z 10-bitnim analogno digitalnim pretvornikom MCP3008. Se pa lahko ti podatki pri drugih tipih krmilnih enot razlikujejo. Analogni priključki se uporabljajo za branje vrednosti tipal, digitalni pa za krmiljenje in preverjanje statusa ostalih naprav.

3.3.1 Branje stanja naprave

Vsako napravo, priključeno v sistem, lahko določimo tako, da nam vrača stanje. Izvedbo priklopa si poglejmo na primeru luči, ki jo krmilimo s klasičnim stikalom. Preko sistema pametne hiše želimo nadzorovati stanje naprave. Luč ima lahko dve stanji - prižgana ali pa ugasnjena. V vezavi to napravimo tako, da vzporedno z lučjo povežemo rele (Slika 3.5), ki bo priključen na en vhodni priključek krmilne enote in nam bo sporočal stanje luči.



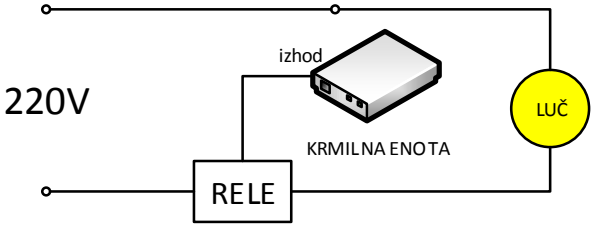
Slika 3.5: Shema nadzorovanja naprave – luči.

Da bi tako napravo lahko vnesli v sistem, moramo določiti tudi nalogo priključka, ki je priključen na krmilno enoto. V tem primeru je naloga enostavna: sporočati nam mora stanje. Na podlagi te naloge definiramo tako imenovano akcijo, ki nam jo ponuja priključek naprave. Za pregled statusa naprave definiramo akcijo »vrni status naprave«, ki jo predstavimo s kodo GET_LIGHT_STATE.

Temu, kar je priključeno na krmilno enoto, sedaj lahko rečemo naprava, ki ima en priključek, na katerem ima eno akcijo: GET_LIGHT_STATE, predstavlja pa nam stanje luči. Podatke takšne, kot smo jih specificirali, vnesemo v podatkovno bazo.

3.3.2 Krmiljenje naprave

Specificiranje naprave, ki jo bomo lahko krmilili, ni nič težje od preverjanja stanja naprave v prejšnjem podpoglavju. Za boljšo razumljivost navedimo primer, ko imamo v hiši klasično luč, ki jo želimo krmiliti preko sistema. Klasično stikalo zamenjamo z električnim stikalom oziroma relejem (Slika 3.6), ki ga lahko krmilimo s krmilno enoto. Stikalo je priključeno na en izhodni priključek. Določiti moramo samo še naloge, ki jih ima sedaj ta naprava. Njena osnovna naloga je »spremeniti status naprave«. Na podlagi te naloge specificiramo akcije. Akcija za spreminjanje bi bila »spremeni status naprave«. Poimenujemo jo SET_SWITCH_STATE. Da bi lahko prebirali tudi status te naprave, lahko specificiramo še eno akcijo, »preberi status naprave«, zapisano s kodo GET_SWITCH_STATE.



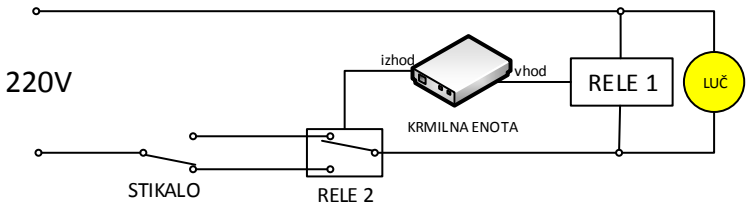
Slika 3.6: Shema krmiljenja luči z električnim stikalom.

Takemu, priključku na vhodno izhodni priključek lahko rečemo naprava, ki ima en priključek in na tem priključku dve akciji, kot smo določili. Podatke nato takšne, kot smo jih specificirali, vnesemo v podatkovno bazo.

3.3.3 Združena naprava

Če zgornja primera združimo, dobimo vezavo, kot jo prikazuje slika 3.7. Taka vezava nam predstavlja povezavo luči kot končne naprave v sistem. Če združimo obe specifikaciji prejšnjih primerov, ima naprava 2 priključka.

Rele 1 nam služi samo za branje vrednosti statusa naprave, rele 2 pa za nastavljanje vrednosti preklopnega stikala, s katerim krmilimo stanje luči: ali je prižgana ali ugasnjena.



Slika 3.7: Shema združene naprave.

En priključek nam bo sporočal stanje luči, ki ima akcijo GET_LIGHT_STATE, drugi priključek pa bo namenjen spreminjanju stanja z akcijama SET_SWITCH_STATE in GET_SWITCH_STATE. V naš sistem to lahko vnesemo kot eno napravo: luč s stikalom ali pa dve ločeni napravi: prva naprava bi bila luč, druga pa stikalo.

Prednost take vezave je tudi, da lahko napravo krmilimo tudi preko klasičnega križnega ali preklopnega stikala neodvisno od pametnega sistema.

Podobno kot smo definirali končno napravo luč, lahko definiramo različne naprave. To so lahko razna tipala, vremenske postaje, stikala, okna, vrata … Kompleksnejše naprave bi lahko bile združene iz več posamičnih. Kot eno napravo bi lahko prikazali in priključili tudi celotne sisteme, kot so centralni vir ogrevanja, centralni sesalni sistem, prezračevalni sistem…

Kako podatke naprav programsko podpreti, si bomo ogledali v predstavitvi centralnega strežnika, ko si bomo ogledali programske razrede, ki predstavljajo končne naprave.

3.4 Zasnova podatkovnega modela

Podatkovni model bomo uporabili za predstavitev konceptualnega in zunanjega nivoja podatkovne baze. Za načrtovanje podatkovnega modela je pomembno dobro razumevanje arhitekturnega modela, poleg slednjega je pomembno tudi poznavanje krmilnih enot in tipov končnih naprav, ki so priključene v sistem.

Pri predstavitvi naprav pametnega sistema smo spoznali vrste naprav v sistemu, pri priključevanju končnih naprav pa je pomembno predvsem, kako napravo priključiti na krmilno enoto in pripraviti podatke za vnos v podatkovno bazo. Poleg vseh informacij o napravah, ki so v sistemu, končnih napravah, ki so priključene na krmilne enote, potrebujemo tudi informacije o opravilih, ki jih tak sistem izvaja, navideznih napravah in o grafični predstavitvi elementov. Hranjenje informacij o grafični predstavitvi elementov je predvsem pomembno pri dinamično zgrajenem uporabniškem grafičnem vmesniku.

Na osnovi ideje smo najprej pripravili E-R-diagram. Na podlagi le-tega pa relacijski model, ki je bil osnova za pripravo relacijske baze. Relacijski model je zasnovan tako, da vsebuje podatke za tri različna področja:

- podatke o krmilnih enotah in napravah, ki so priključene v sistem - vključno z vsemi vhodno/izhodnimi priključki,
- podatke o grafičnih elementih na grafičnem vmesniku,
- podatke, potrebne za delo z dogodki.

3.4.1 Relacijski model

Vsaka tabela v podatkovni bazi vsebuje naslednje podatke:

- ID zapisa – polju ID je dodeljen podatkovni tip število (»*AvtoNumber*«), kar pomeni, da pri vpisu sistem sam ustvarja novo enolično število ID za vsak nov zapis. Na polju ID imamo postavljen tudi primarni ključ, ki nam nedvoumno identificira vsak zapis, ki je shranjen v tabeli. ID se uporablja za vse relacije znotraj sistema.
- GUID – polju GUID je dodeljen podatkovni tip GUID, kar pomeni globalni enolični identifikator. Je 128-bitno število, ki ga generira Windows operacijski sistem. GUID se uporablja za izvoz/uvoz podatkov v sistem. Pri prenosu podatkov na drugi sistem se ID-ključ spreminja, GUID pa ostaja enak in s tem vedno nedvomno predstavlja določen zapis.

Relacijski model je razdeljen na tri sklope. Prvi sklop relacijskega modela (Slika 3.8) vsebuje podatke o krmilnih enotah in napravah, ki so priključene v sistem, vključno s podatki o vhodno/izhodnih priključkih:

Tabela *homeSystem* vsebuje poleg ID-ja in GUID-ja tudi zapis o imenu in opisu našega sistema. Služi nam predvsem za izbiro sistema. V praksi bi lahko eno podatkovno bazo uporabili za več sistemov. Relacije v tabeli:

- sistem ima lahko 0 do n kategorij, ena kategorija pa pripada natanko enemu sistemu,
- sistem ima lahko 0 do n strežnikov za proženje dogodkov, en strežnik za proženje dogodkov pa pripada natanko enemu sistemu,
- sistem ima lahko 0 do n krmilnih enot, ena krmilna enota pa pripada natanko enemu sistemu.

Tabela *Reg_homeControllerType* vsebuje šifrant tipov krmilnikov: vsebuje podatke o imenu in opisu tipa krmilnika. To so lahko Raspberry Pi, Arduino, PC … Relacije v tabeli:

- tip krmilnika ima lahko 0 do n krmilnih enot, ena krmilna enota je lahko natanko enega tipa.

Tabela *homeController* vsebuje podatke o krmilnih enotah, ki so registrirane v sistemu. Vsebuje podatke o sistemu, ki mu pripada, o tipu, TCP/IP-naslov, port ter ime in opis. Služi nam za kreiranje programskega razreda v našem sistemu. Relacije v tabeli:

- krmilna enota ima lahko 0 do n vhodno/izhodnih priključkov, en priključek pripada natanko eni krmilni enoti,
- krmilna enota ima lahko 0 do n priključenih naprav, ena naprava pripada natanko eni krmilni enoti.

Tabela *homeControllerPin* vsebuje podatke o vhodno/izhodnih priključkih. Za vsak priključek tako vsebuje podatke o krmilni enoti, ki ji pripada, naslov vhodno/izhodnega priključka ter ime in opis.

Tabela *Reg_homeDeviceType* je tabela s šifrantom tipov naprav. Vsebuje podatke o imenu in opisu tipa naprave. To so luč, stikalo, okno … Relacije v tabeli:

- tip naprave ima lahko 0 do n naprav, ena naprava je lahko natanko enega tipa.

Tabela *homeDevice* vsebuje podatke o končnih napravah, ki so priključene v naš sistem. Vsebuje podatke o vrsti krmilne enote, na katero je priključena, kakšnega tipa je ter ime in opis. Služi nam za kreiranje programskega razreda naprave. Pred vnosom posamezne naprave v sistem moramo imeti predvsem dobro specificirano, kakšni so njeni vhodno/izhodni priključki ter na katere vhodno/izhodne priključke krmilnika je priključena. Relacije v tabeli:

- naprava ima lahko 0 do n vhodno/izhodnih priključkov, en priključek pripada natanko eni napravi.

Tabela *Reg_homeDevicePinType* je tabela s šifrantom tipov vhodno/izhodnega priključka. Vsebuje podatke o imenu in opisu tipa priključka. To je lahko analogni, digitalni, vhodni ali izhodni priključek. Podatke potrebujemo predvsem za branje vrednosti iz naprave. Relacije v tabeli:

- tip priključka ima lahko 0 do n priključkov, en priključek je lahko natanko enega tipa.

Tabela *homeDevicePin* vsebuje podatke o vhodno/izhodnih priključkih naprave: kateri napravi pripada, na kateri vhodno/izhodni priključek je priključek naprave priključen, kakšnega tipa je ter ime in opis priključka. Relacije v tabeli:

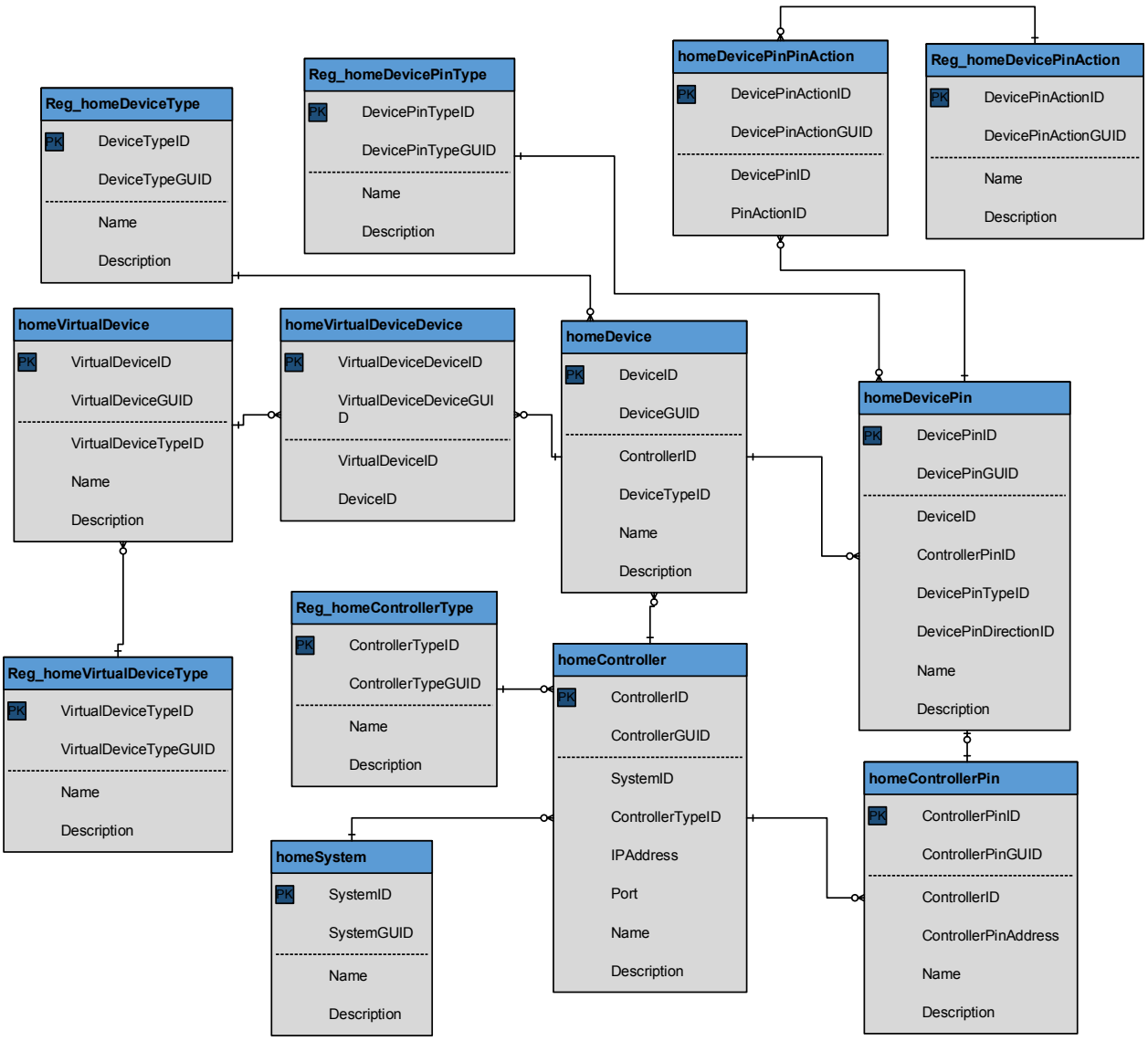
- vhodno/izhodni priključek naprave je priključen na natanko en vhodno/izhodni priključek krmilne enote,
- vsak vhodno/izhodni priključek naprave ima lahko 0 do n akcij.

Tabeli *Reg_homeDevicePinAction* in *homeDevicePin* sta povezani preko relacijske tabele *homeDevicePinPinAction*, ker je med njima relacija n do n. Relacije v tabeli:

- en vhodno/izhodni priključek ima lahko n akcij, ena akcija lahko pripada n vhodno/izhodnim priključkom.

Tabela Reg*_homeDevicePinAction* je tabela s šifrantom akcij vhodno/izhodnih priključkov. Vsebuje podatke o imenu in opisu akcije. To so akcije, ki smo jih spoznali pri določanju končnih naprav: GET_LIGHT_STATE, GET_SWITCH_STATE, SET_SWITCH_STATE …

Tabele *homeVirtualDevice*, *Reg_homeVirtualDeviceType* in *homeVirtualDeviceDevice* so zasnovane za virtualne naprave, ki bi jih imeli v sistemu. Primer virtualne naprave bi bil stikalo za izklop vseh luči v hiši. Preko relacijske tabele *homeVirtualDeviceDevice* sta povezani tabeli *homeDevice* in *homeVirtualDevice,* tako da ima virtualna naprava dodeljene naprave, s katerimi bo delala.



Slika 3.8: Sklop rel. modela, ki zajema podatke o končnih napravah in krmilnih enotah.

Drugi del relacijskega modela (Slika 3.9) vsebuje podatke o grafičnih elementih. Na podlagi teh podatkov bomo dinamično zgradili naprave na aplikaciji za upravljanje in nadzorovanje sistema.

Tabela *homeUIElement* vsebuje referenco, kateri element predstavlja. To je lahko dogodek, naprava, krmilna enota, strežnik za proženje ali pa virtualna naprava. Relacije v tabeli:

- grafični element lahko predstavlja natančno eno napravo, ena naprava pa je lahko predstavljena z natanko enim grafičnim elementom.

Tabela *homeUIElementType* vsebuje podatke o tipu referenčnega elementa. To je lahko dogodek, naprava, krmilna enota, strežnik za proženje ali pa virtualna naprava. Relacije v tabeli:

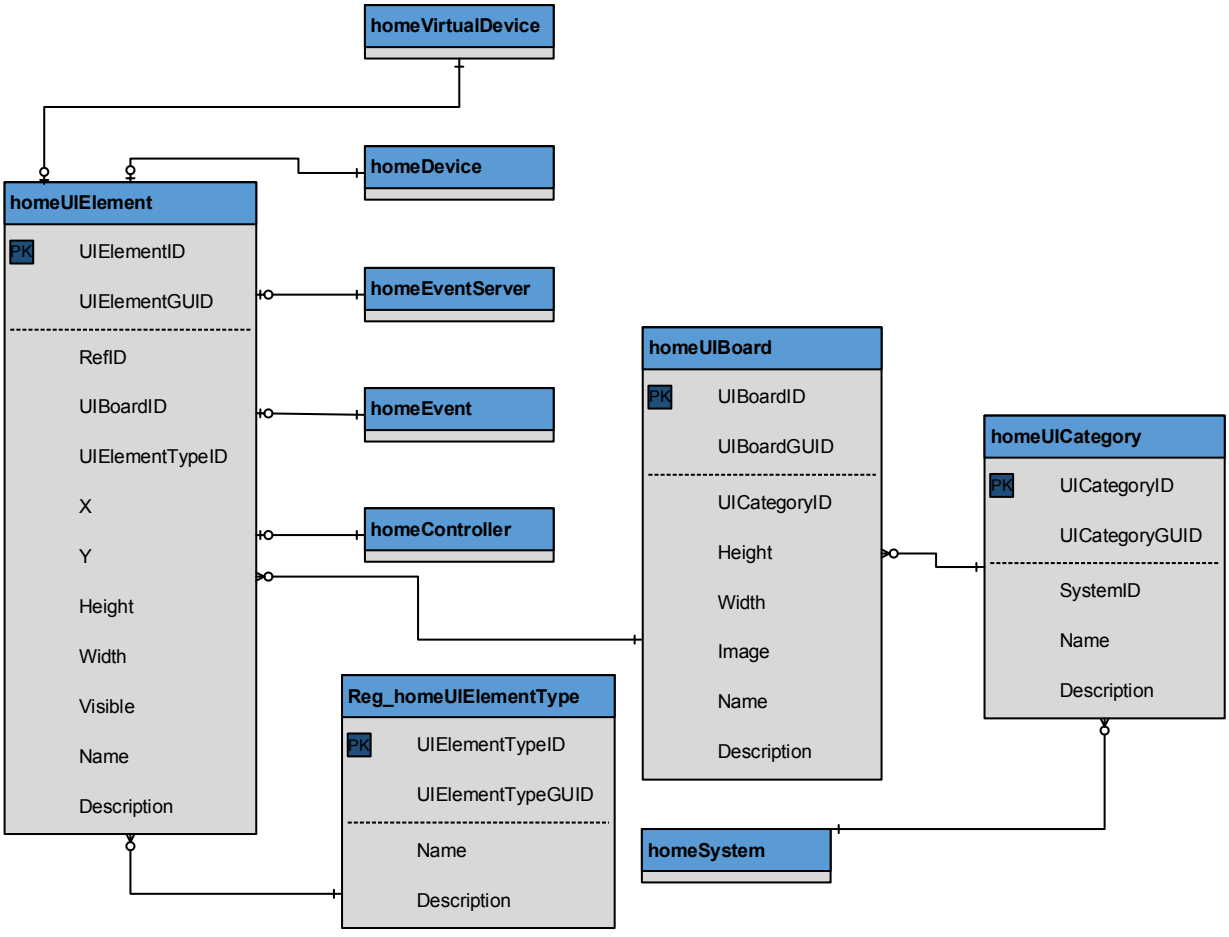
- tip grafičnega elementa ima lahko 0 do n grafičnih elementov, en grafični element lahko predstavlja natanko en element.

Tabela *homeUIBoard* vsebuje podatke o osnovnih ploščah. To so predvsem panele v grafičnem vmesniku, na katerem so grafični elementi. Relacije v tabeli:

- ena osnovna plošča ima lahko 0 do n grafičnih elementov, en grafični element lahko pripada natanko eni plošči.

Tabela *homeUICategory* vsebuje podatke o kategorijah. To je lahko kuhinja, kopalnica, mansarda … Relacije v tabeli:

- ena kategorija ima lahko 0 do n osnovnih plošč, ena plošča pripada natanko eni kategoriji,
- ena kategorija lahko pripada natanko enemu sistemu.



Slika 3.9: Sklop rel. modela, ki zajema podatke o grafičnih elementih.

Tretji sklop relacijskega modela je zasnovan za avtomatsko izvajanje opravil. To so relacijske tabele *HomeEventServer*, *homeEvent*, *homeEventScheduler*, *Reg_homeEventStatus*, *homeEventType* in relacijske tabele *homeEventVirtualDevice*, *homeEventDevice* ter *homeEventController*. V ospredju je ideja o strežniku v sklopu centralne enote, ki bi po koledarju ali ciklično izvajal določene naloge. Npr.: nekemu dogodku smo dodelili več naprav. To sta recimo klasična vrtna luč in svetlobno tipalo. Takemu dogodku bi določili pravilo »Če je vrednost svetlobnega tipala pod določeno mejo, potem prižgi luč, sicer jo ugasni.« Dogodek bi se izvajal ciklično na 10 minut, vrtna luč bi se vsako noč avtomatsko prižigala, podnevi pa bi bila ugasnjena.

3.4.2 Začetna vsebina podatkovne baze

Za pravilno delovanje centralnega strežnika in spletne aplikacije potrebujemo v podatkovni bazi izpolnjene naslednje podatke: tip krmilne enote, tip naprave, tip vhodno/izhodnih priključkov … To so predvsem razni šifranti. Teh podatkov ni mogoče spreminjati na aplikaciji. V primeru, da bi se izkazalo, da je podatke potrebno spremeniti ali pa jim dodati novega, je potrebno to storiti v podatkovni bazi. Vendar moramo biti previdni, saj teh podatkov nikoli ne smemo brisati ali zamenjevati vrstnega reda, ker morajo biti usklajeni z razredom »enum« v programski kodi.

Podatki, ki so že vnaprej vneseni, so zapisani v naslednjih tabelah:

- *Reg_homeDevicePinType* je tabela, v kateri so vnaprej vneseni tipi vhodno/izhodnih priključkov končnih naprav: ANALOG_INPUT, ANALOG_OUTPUT, DIGITAL_INPUT, DIGITAL_OUTPUT …,
- *Reg_homeDevicePinAction* je tabela, v kateri so podprte akcije, ki se lahko izvajajo na priključkih: GET_LIGHT_STATE, GET_SWITCH_STATE …,
- *Reg_homeVirtualDeviceType* je tabela, v kateri so tipi virtualnih naprav: GROUPE_SWITCH, GROUPE_TEMPERATURE_SENSOR, CUSTOM_VIRTUAL _DEVICE …,
- *Reg_homeUIElementType* je tabela, v kateri so tipi grafičnih elementov za prikaz v spletni aplikaciji: DEVICE, CONTROLLER, VIRTUAL_DEVICE …,
- *Reg_homeEventStatus* je tabela, v kateri so predvideni statusi dogodkov: STOP, PAUSE, RUN …,
- *Reg_homeEventType* je tabela, v kateri so tipi dogodkov, ki jih centralni strežnik lahko izvaja: PERMANENT, ONE_TIME, SCHEDULER …,

- *Reg_homeControllerType* je tabela, v kateri so vnaprej vneseni tipi krmilnih enot (Slika 3.10), ki so podprte v sistemu:

	ControllerTypeID	ControllerTypeGUID	Name	Description
1	1	D2D03A2D-375F-408E-A233-4BC9DEE9131F	RASPBERRYPI	NULL
2	2	F535E2BB-0B8F-4F77-86F2-1988AB9CA5FF	ARDUINO	NULL
3	3	7235ECEB-DC6D-4B6A-BC2F-A28213AB8F43	PC	NULL

Slika 3.10: Zapis tipov krmilnih enot v podatkovni bazi.

Na podlagi zapisa v tabeli se preko skript samodejno generira razred »enum« (Slika 3.11), ki ga uporabimo pri implementaciji centralnega strežnika:

```
namespace SmartHome.Types.Enums
{
    public enum ControllerType
    {
        RASPBERRYPI=1,
        ARDUINO,
        PC
    }
}
```

Slika 3.11: Razred »Enum« s tipi krmilnih enot.

- *Reg_homeDeviceType* je tabela, v kateri so vnaprej vneseni tipi končnih naprav (Slika 3.12), ki so podprti v sistemu:

	DeviceTypeID	DeviceTypeGUID	Name	Description
1	1	2F823E38-F4F7-4EC5-B0CE-4CD10DD85068	SWITCH	Preklopno stikalo
2	2	14402E22-CD03-4175-8286-8A823B8ECB9E	LIGHT	Luč
3	3	94D1E36E-72B7-4517-8527-908348C66031	VENTILATOR	Ventilator
4	4	69250F33-E84E-49F4-8332-E0E2D7EBBD08	MOTION_SENSOR	Senzor gibanja
5	5	168F5EA4-58F1-475C-A641-FCDC5A85C4AC	TEMPERATURE_SENSOR	Senzor za merjenje temperature

Slika 3.12: Zapis tipov naprav v bazi.

Na podlagi zapisa v tabeli se preko skript samodejno generira razred »enum« (Slika 3.13), ki ga uporabimo pri implementaciji centralnega strežnika:

```
namespace SmartHome.Types.Enums
{
    public enum DeviceType
    {
        SWITCH = 1,
        LIGHT,
        VENTILATOR,
        MOTION_SENSOR,
        TEMPERATURE_SENSOR
    }
}
```

Slika 3.13: Razred »Enum« s tipi končnih naprav.

Tako so v podatkovni bazi zapisani vsi šifranti. Zelo pomembno je, da sta razred »enum« in zapis v podatkovni bazi usklajena po ID-ju. Predvsem pa je pomembno, da na stolpcu, ki predstavlja ID, ne uporabljamo avtomatskega povečevanja ID-ja ob vnosu takih šifrantov v podatkovno bazo.

Če bi želeli podpreti nov tip končne naprave ali novo krmilno enoto, je potrebno dopolniti tako razred »enum« kot tudi zapis v podatkovni bazi

Poglavje 4 Implementacija porazdeljenega sistema pametne hiše

Na podlagi zasnovanega sistema pametne hiše smo izdelali programske komponente, ki so potrebne za delovanje sistema. Pri implementaciji programskih modulov sistema smo bili predvsem pozorni na splošnost, modularnost in dinamičnost sistema. Programski modul krmilne enote mora biti čimbolj preprost in splošen, da ga lahko uporabimo na večini kartičnih ali splošnih računalnikov. Za programski modul centralne enote pa je pomembno, da zasnujemo pravilno arhitekturo, ki nam bo omogočala njegov hiter in fleksibilen razvoj. Skupaj s spletno aplikacijo smo uporabili n-tirno arhitekturo aplikacije. Pomembno je tudi, da izberemo pravi programski jezik, ki nam bo omogočal hitrejšo in lažjo implementacijo. V nadaljevanju si bomo najprej pogledali, katero programsko in strojno opremo potrebujemo za razvoj takega sistema, nato pa bomo bolj podrobno pogledali implementacijo podatkovne baze, programski modul centralnega strežnika, spletne aplikacije in krmilne enote.

4.1 Uporabljena strojna in programska oprema

Za implementacijo sistema smo uporabili najrazličnejšo strojno in programsko opremo. Za strojno opremo smo uporabili kar domač stacionarni računalnik in kartični računalnik Raspberry Pi. Zelo podrobno smo spoznali tudi različna integrirana orodja za razvijanje programske opreme, ki jih bomo v nadaljevanju predstavili.

4.1.1 Strojna oprema

Za razvoj vseh in testiranje določenih programskih sklopov smo uporabili namizni računalnik Intel Core I5, ki ima naslednje lastnosti:

- procesor Intel Core I5 2,67 GHz,
- 12 GB DDR3 pomnilnika,
- grafično kartico Asus R5 230 series,
- SATA trdi disk WD320AAKS.

Operacijski sistem, na katerem smo razvijali in testirali, je bil Windows Server 2012.

Za krmilnik pametne hiše smo uporabili kartični računalnik Raspberry Pi [5] Model B+ s 512MB pomnilnika RAM na sliki 4.1.

Operacijski sistem, ki smo ga namestili na kartični računalnik, je bil Linux RASPBIAN verzije 3.12.

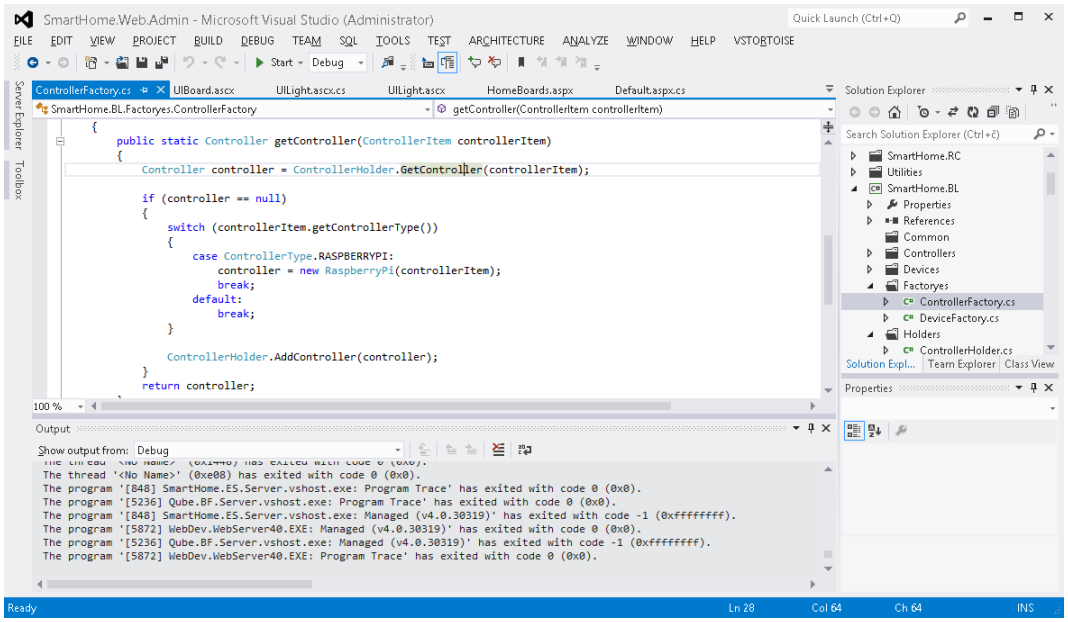
Slika 4.1: Raspberry Pi Model B+ 512MB [5].

4.1.2 Programska oprema

Za implementacijo porazdeljenega sistema smo uporabili različna programska orodja, ki so pripomogla predvsem k hitrejšemu in lažjemu razvoju programskih modulov:

- Visual studio 2012

  Visual studio [8] je integrirano razvojno okolje (IDE) (Slika 4.2), ki omogoča razvoj aplikacij v programskem jeziku C, C#, C++, Visual Basic, JavaScript, Visual F#, … Podpira tudi spletne programske jezike XML, XSLT, HTML, XHTML in CSS, Omogoča razvoj spletnih aplikacij, servisov, aplikacij z grafičnim vmesnikom ali brez, aplikacij za pametne telefone z operacijskim sistemom Windows, … . Urejevalnik kode podpira obarvanje kode, zaključevanje kode, prikaz seznamov metod, spremenljivk, funkcij … Vključuje tudi vrsto pomožnih orodij kot so: Spy++, Trace tool, Create GUID, …



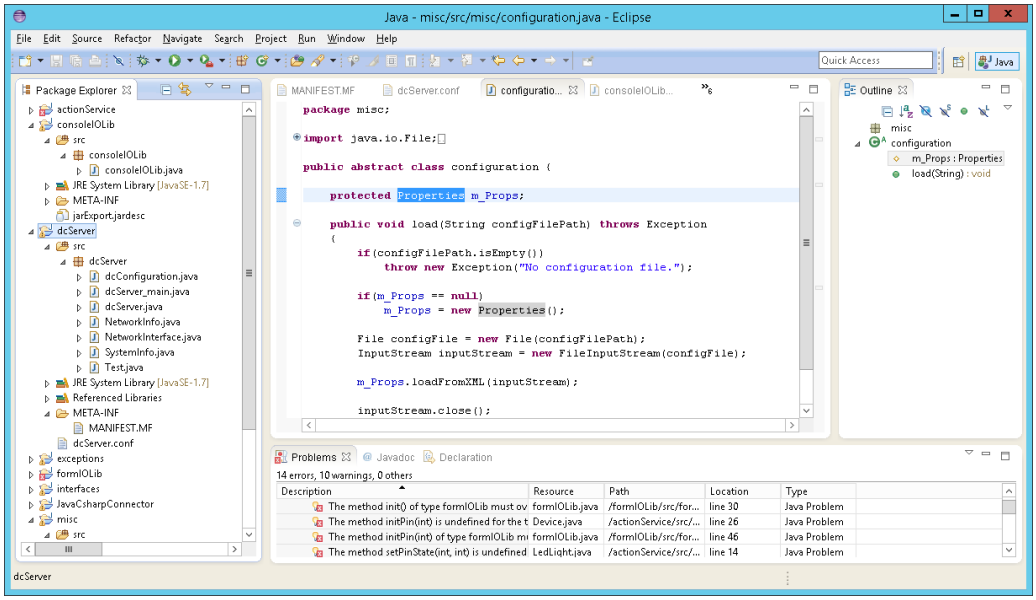
Slika 4.2: Integrirano razvojno okolje Visual Studio (IDE).

- Eclipse

Eclipse [9] je integrirano razvojno okolje (IDE) (Slika 4.3), ki omogoča razvoj aplikacij v programskem jeziku Java, JavaScript, C, C++ … in je na voljo za vse operacijske sisteme. Na voljo ima veliko uporabnih razširitev, izmed katerih smo uporabljali naslednje:

- »Fat jar« dodatek za ustvarjanje jar [11] datotek,
- »Subversion« klient za beleženje sprememb za izvedbo na lokalni SVN-strežnik,
- »Easy Shell« za zagon urejevalnika datotek, odpiranje »bash« lupin,
- »Texlipse« za pisanje programskih modulov,
- »Quick Search« za hitrejše iskanje,
- »Eclipse Color Theme« za lažje branje programske kode.

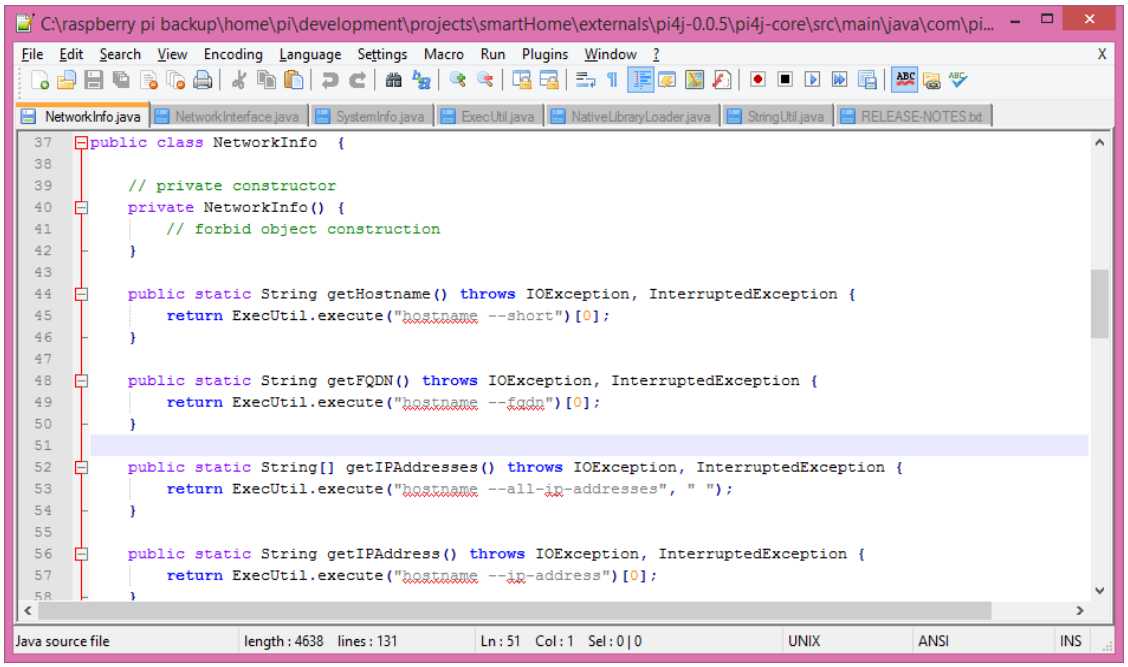
Tako smo imeli v istem orodju združenih veliko uporabnih razširitev in funkcionalnosti.



Slika 4.3: Integrirano razvojno okolje Eclipse IDE.

- Notepad++

Notepad++ [10] (Slika 4.4) je odprtokodni tekstovni urejevalnik, ki podpira več programskih jezikov. Deluje v okolju Microsoft Windows.



Slika 4.4: Tekstovni urejevalnik Notepad++.

4.2 Implementacija podatkovne baze

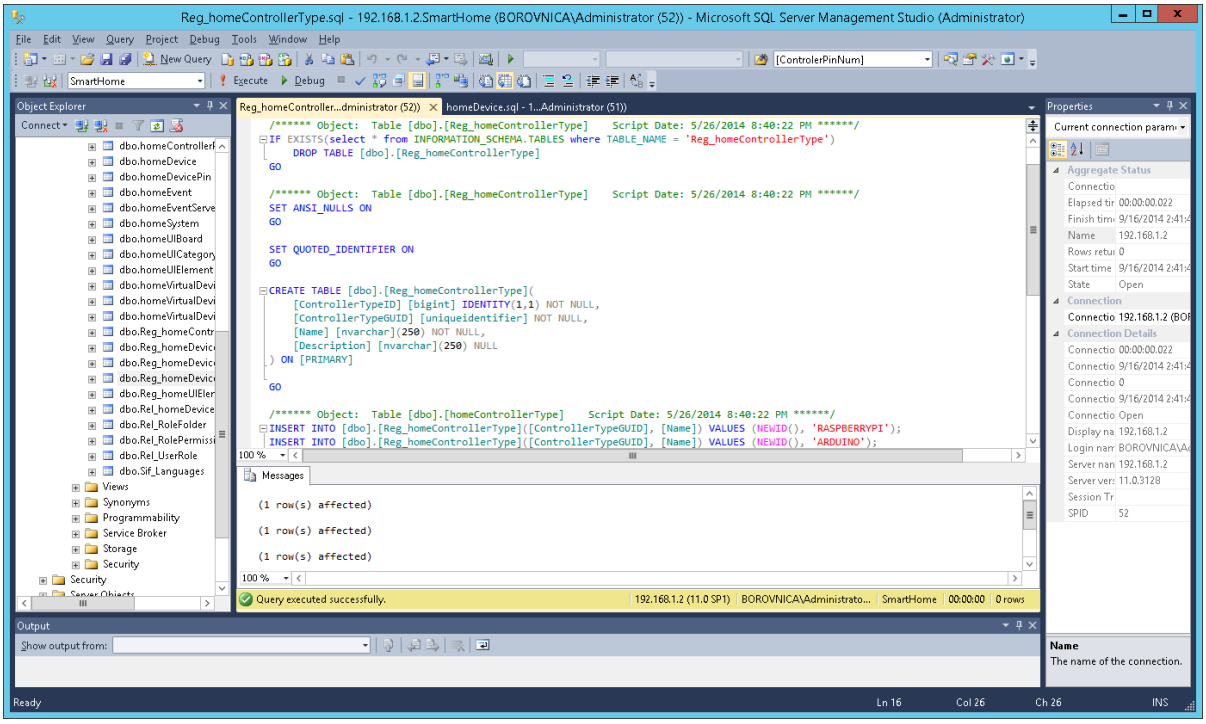
Na podlagi zasnovanega relacijskega modela smo izdelali podatkovno bazo. Za vse tabele podatkovne baze smo s stavki SQL spisali v skriptah SQL, kamor smo definirali tudi testne podatke. Primer skripte SQL za kreiranje tabele *homeDevice*, ki hrani podatke o napravah prikazuje Slika 4.5.

```
USE [SmartHome]
GO

/*brisanje obstoječe tabele*/
IF EXISTS(select * from INFORMATION_SCHEMA.TABLES where TABLE_NAME = 'homeDevice')
       DROP TABLE [dbo].[homeDevice]
GO

/*ustvarjanje tabele*/
CREATE TABLE [dbo].[homeDevice](
       [DeviceID] [bigint] IDENTITY(1,1) NOT NULL,
       [DeviceGUID] [uniqueidentifier] NOT NULL,
       [ControllerID] [bigint] NOT NULL,
       [DeviceTypeID] [bigint] NOT NULL,
       [Name] [nvarchar](250) NOT NULL,
       [Description] [nvarchar](250) NULL
) ON [PRIMARY]

GO

/*predizpolnjeni podatki podatki*/
DECLARE @ControllerID bigint = 1; /*Controller 1*/
DECLARE @DeviceTypeID bigint = 1; /*Preklopno stikalo*/
INSERT INTO [dbo].[homeDevice]([DeviceGUID], [ControllerID], [DeviceTypeID], [Name])
VALUES (NEWID(), @ControllerID, @DeviceTypeID, 'Stikalo_1');
INSERT INTO [dbo].[homeDevice]([DeviceGUID], [ControllerID], [DeviceTypeID], [Name])
VALUES (NEWID(), @ControllerID, @DeviceTypeID, 'Stikalo_2');
SET @DeviceTypeID = 2; /*Luč*/
INSERT INTO [dbo].[homeDevice]([DeviceGUID], [ControllerID], [DeviceTypeID], [Name])
VALUES (NEWID(), @ControllerID, @DeviceTypeID, 'Luč_1');
INSERT INTO [dbo].[homeDevice]([DeviceGUID], [ControllerID], [DeviceTypeID], [Name])
VALUES (NEWID(), @ControllerID, @DeviceTypeID, 'Luč_2');
SET @ControllerID = 2; /*PC*/
SET @DeviceTypeID = 1; /*Preklopno stikalo*/
INSERT INTO [dbo].[homeDevice]([DeviceGUID], [ControllerID], [DeviceTypeID], [Name])
VALUES (NEWID(), @ControllerID, @DeviceTypeID, 'Stikalo_3');
GO
```

Slika 4.5: Skripta SQL za ustvarjanje tabele končnih naprav.

V vse skripte smo na začetku dodali brisanje obstoječe tabele. To nam pride prav pri večkratnem izvajanju skripte SQL. Nato z ukazom »CREATE TABLE« ustvarimo tabelo. Na koncu so v skripti dodani še stavki za napolnitev testnih podatkov, ki jih potrebujemo za razvoj ostalih programskih modulov.

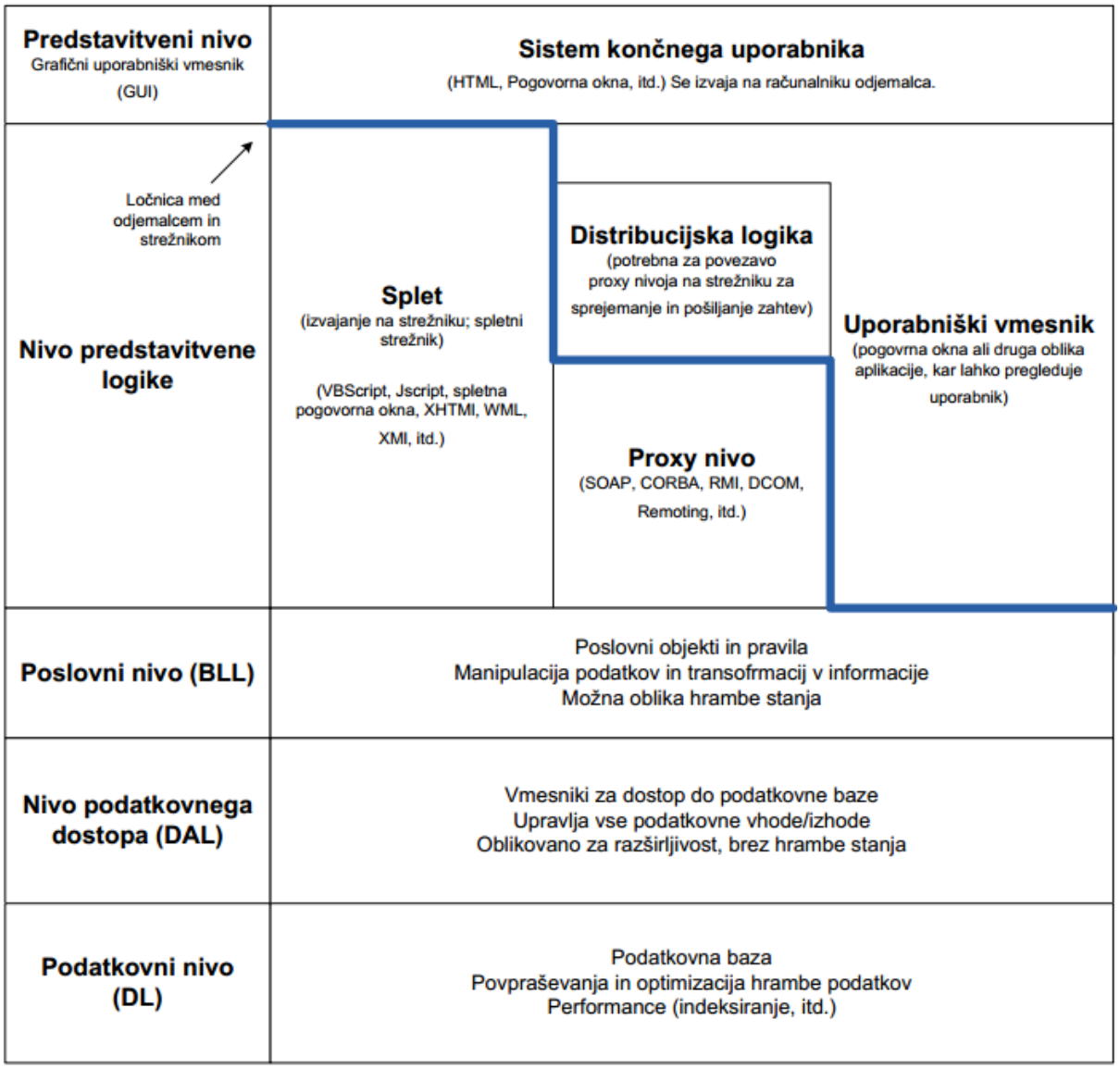
Vse skripte izvedemo s pomočjo programske opreme Microsoft SQL Management Studio (Slika 4.6). Skripte SQL uporabljamo tudi za arhiviranje podatkov iz podatkovne baze.



Slika 4.6: Izvedba skript SQL z orodjem MS SQL Management Studio.

4.3 Centralni strežnik

Pravilna implementacija centralnega strežnika je zelo pomembna predvsem z vidika vzdrževanja in nadgradenj. Zato smo pri implementaciji upoštevali, da mora biti strežnik zgrajen čimbolj modularno in dinamično. Tako izgradnjo strežnika nam omogoča n-tirna arhitektura, kot prikazuje slika 4.7. Poleg modularnosti nam omogoča tudi fleksibilen razvoj aplikacije. Sistem je tako razgrajen na nivoje, kar pomeni, da nadgradnja ne pomeni spreminjanja celotne aplikacije.



Slika 4.7: N-tirna arhitektura izgradnje centralnega strežnika [7].

V našem sistemu centralni strežnik imenujemo »coreServer«. Ker je »coreServer« razvit v programskem jeziku C#, moramo za centralno strežniško enoto uporabiti strežnik z operacijskimi sistemom Windows. ali drug sistem, kjer je nameščeno okolje »Mono«.

Za razvoj centralnega strežnika v programski jezik C# smo se odločili iz več razlogov:

- odprte opcije, dobra podpora .net-programskega jezika,
- hitrejši razvoj,

- možnost lepšega pisanja »get« in »set« metod, ki odlično nadomeščajo siceršnje pisanje takih metod,

- možnost uporabe knjižnice »Jayrock« za JSON-RPC oddaljene klice,

- zelo pomembno pa je, da je lahko spletna aplikacija razvita v programskem jeziku .net in tako skupaj s centralnim strežnikom uporabljata iste skupne razrede, ki si jih lahko med seboj tudi izmenjujeta. Razred lahko z enkratnim implementiranjem nato uporabljaš v katerem koli delu aplikacije …

4.3.1 Podatkovni nivo

V našem sistemu je podatkovni nivo naš podatkovni model oziroma podatkovna baza, v kateri imamo vse tabele iz podatkovnega modela (Slika 3.8 in 3.9). Poleg teh so v podatkovni bazi zasnovane tudi tabele za hranjenje podatkov o opravilih. To so tabele: *HomeEventServer*, *homeEvent*, *homeEventScheduler*, *Reg_homeEventStatus*, *homeEventType* in relacijske tabele *homeEventVirtualDevice*, *homeEventDevice* in *homeEventController*. V podatkovno bazo smo dodali tudi posamične tabele za potrebne avtentikacije, avtorizacije, lokalizacije in dinamično izgradnjo spletne aplikacije. To so tabele: *coreFolders*, *coreFolderType*, *corePermission*, *coreRole*, *coreTheme*, *coreUser*, *coreUserAuthentication*, *coreUserAuthenticationType*, *coreUserSettings*, *Rel_RolePermission*, *Rel_RoleFolder*, *Rel_UserRole* in *Sif_Languages*.

Za vse tabele so v podatkovni bazi zasnovane »store« procedure, preko katerih dostopa nivo podatkovnega dostopa. »Store« procedura je skupek stavkov SQL, ki so shranjeni v podatkovni bazi. Aplikaciji omogočajo dostop do relacijske podatkovne baze.

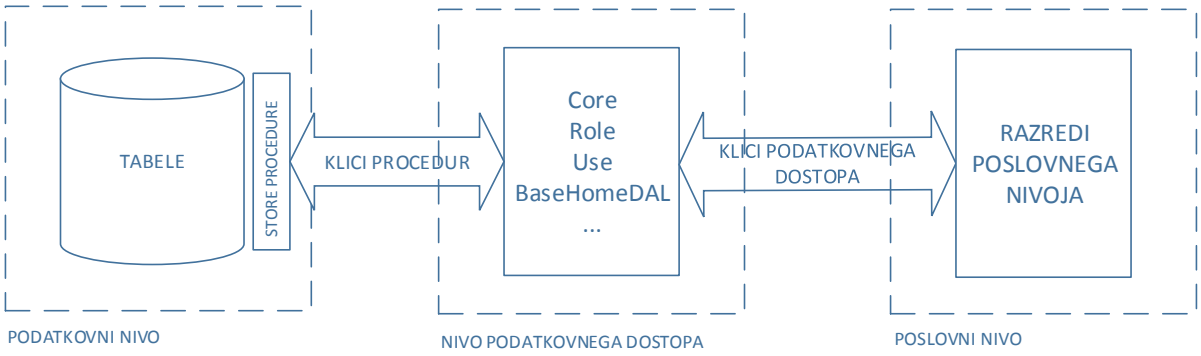
4.3.2 Nivo podatkovnega dostopa

Ta nivo je vmesnik za dostop do podatkov v bazi. Skrbi za povezovanje na podatkovno bazo, klicanje »store« procedur ter ustvarjanju razredov, ki predstavljajo zapis v podatkovni bazi. V našem centralnem strežniku imamo skupni razred »BaseDAL«, ki skrbi za implementacijo povezovanja na podatkovno bazo. Vsi ostali razredi, ki predstavljajo del podatkovnega dostopa, pa morajo podedovati vse metode iz skupnega razreda. To so razredi:

- »Core« in »Role« – razreda skrbita za dostop do podatkov o avtentikaciji, avtorizaciji in dinamični izgradnji spletne aplikacije,

- »User« – razred skrbi za pridobivanje podatkov o uporabnikih sistema pametne hiše,

- »BaseHomeDAL« – razred skrbi za pridobivanje vseh podatkov iz podatkovne baze, o krmilnih enotah, končnih napravah, akcijah, grafičnih podatkov …



Slika 4.8: Prikaz nivoja podatkovnega dostopa.

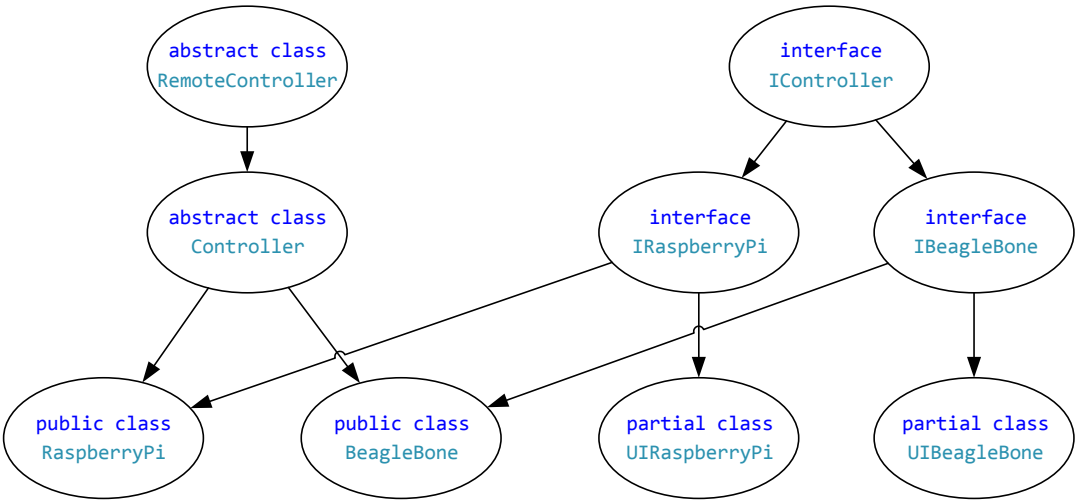
Razredi poslovnega nivoja do podatkov iz podatkovne baze vedno dostopajo preko razredov na nivoju podatkovnega dostopa. Ti razredi pa kličejo »store« procedure na podatkovni bazi. Pri klicih podatkovnega dostopa se vedno kot parameter pošiljajo razredi, ki predstavljajo zapis v podatkovni bazi. Nivo podatkovnega dostopa vedno vrača razrede, ki predstavljajo zapis v podatkovni bazi.

4.3.3 Poslovni nivo

Implementirana je logika samega sistema, upravljanje z razredi naprav, krmilnih enot, izvajanje ustreznih klicev na krmilne enote, obdelovanje podatkov pridobljenih iz krmilne enote, posredovanje podatkov spletni aplikaciji … Predvsem je pomembno, kako so razredi, ki predstavljajo krmilne enote in končne naprave, zgrajeni.

- Zgradba razredov, ki predstavljajo krmilne enote

  V centralnem strežniku je vsaka naprava ali krmilna enota predstavljena z razredi. Kako so arhitekturno zgrajeni razredi v centralnem strežniku in spletni aplikaciji, si lahko pogledamo preko dedovanja le-teh (Slika 4.9). Prikazan primer dedovanja je narejen na dveh različnih tipih krmilnih enot. To sta Raspberry Pi, kateri je predstavljen z razredom »RaspberryPi« in Beagle Bone, kateri je predstavljen z razredom »BeagleBone«.



Slika 4.9: Prikaz zgradbe razredov krmilnih enot.

Centralni strežnik dela z razredi »RaspberryPi« in »BeagleBone«. Spletna aplikacija pa uporablja razreda »UIRaspberryPi« in »UIBeagleBone«, ki sta ASCX-kontroli, vendar dedujeta še druge razrede, ki so del same spletne aplikacije (Slika 4.9).

Abstrakten razred »RemoteController« se »zaveda« krmilne enote, vhodno/izhodnih priključkov ter skrbi za komunikacijo z njo. Do krmilne enote komunikacija poteka preko JSON-RPC oddaljenih klicev, preko katerih »RemoteController« razred kliče metode na programskem modulu krmilne enote. Metoda se samo izvede in vrne rezultate. Za nadaljnjo obdelovanje podatkov skrbi centralni strežnik.

Abstrakten razred »Controller« vsebuje splošne implementacije, ki so skupne vsem razredom, ki predstavljajo kontrolne enote.

Razreda »RaspberryPi« in »BeagleBone« predstavljata končni krmilni enoti in vsak vsebuje svoje specifične implementacije.

Razred »UIRaspberryPi« je naša ASCX-kontrola v spletni aplikaciji, ki grafično predstavlja krmilno enoto.

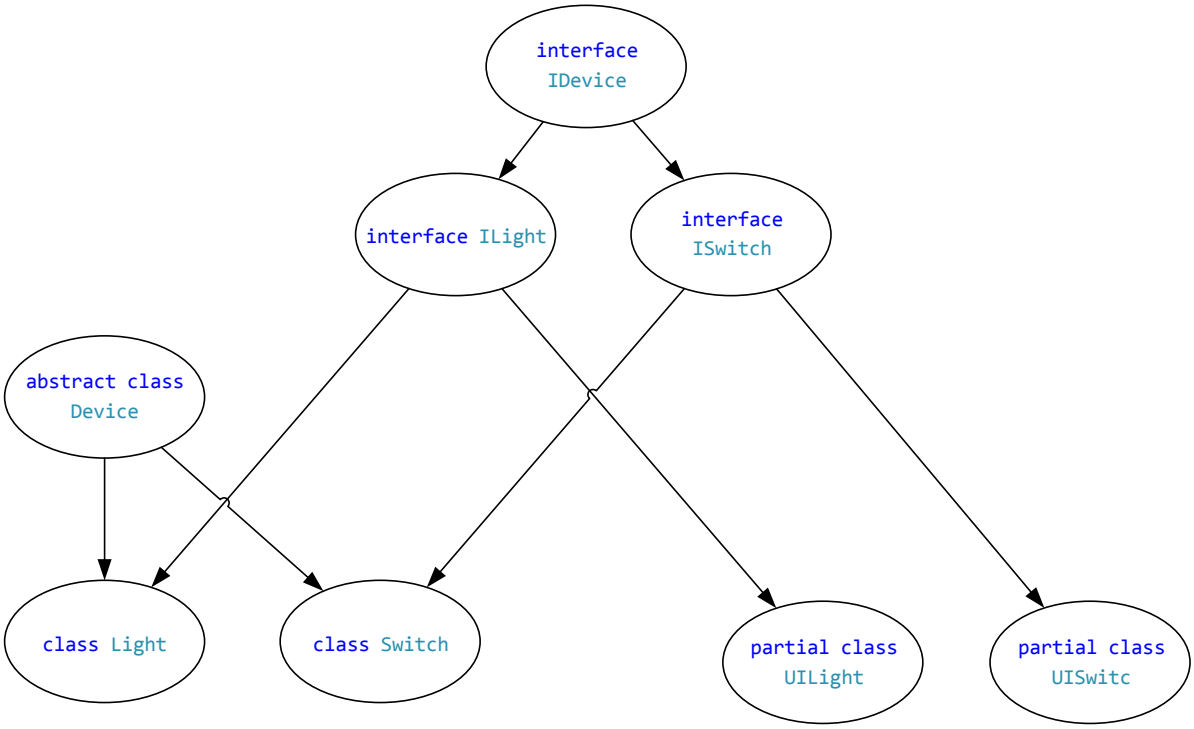
Pri implementaciji imamo »interface«, ker nam predpisujejo, katere metode morajo razredi implementirati.

»IController« je splošni »interface«, ki predpisuje, katere metode morajo vsi razredi krmilnih enot implementirati.

»IRaspberryPi« specifičen »interface« pa predpisuje, katere metode je potrebno implementirati tako na ASCX-kontroli kot tudi v razredu na strežniški strani.

- Zgradba razredov, ki predstavljajo končne naprave

  Spodaj je zelo podobna slika dedovanja (Slika 4.10), le da je brez »Remote« razreda. Dedovanje iz »inteface« razredov imamo tudi tukaj za predpisovanje metod, ki jih moramo implementirati na ASCX-kontrolah spletne aplikacije in na centralnem strežniku.



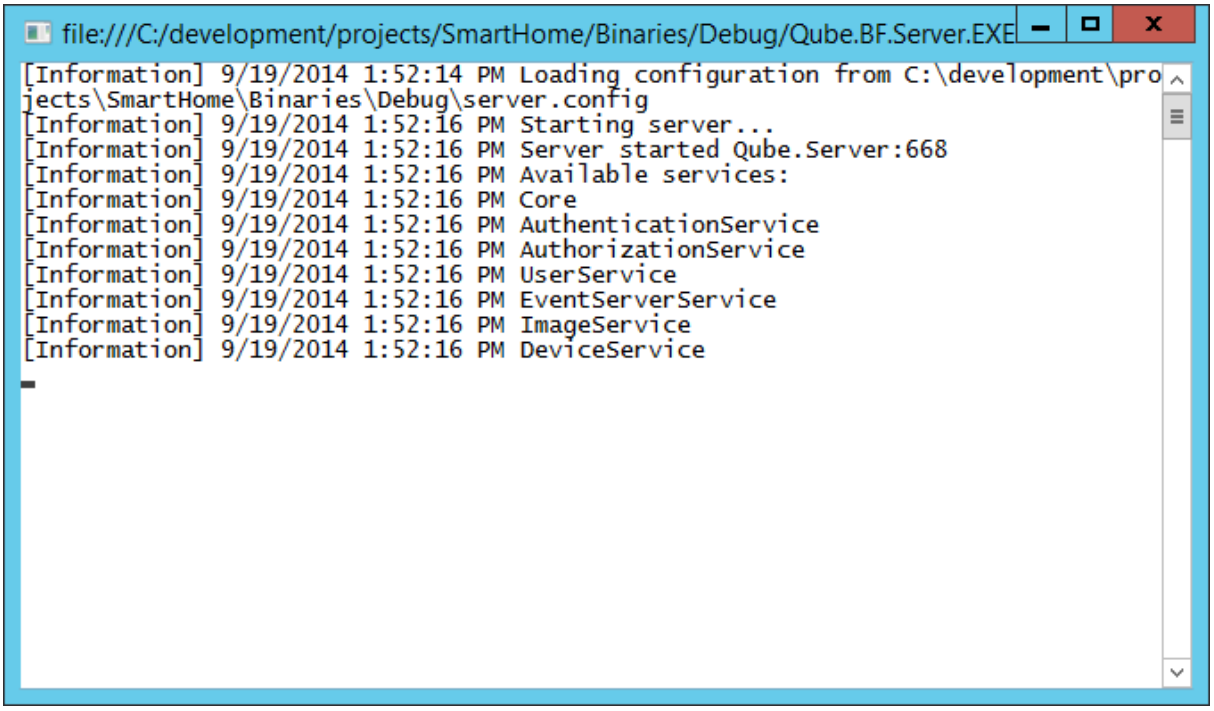
Slika 4.10: Prikaz zgradbe razredov končnih naprav.

Razred naprave za izvedbo akcij uporablja razred, ki predstavlja krmilno enoto, na katero je naprava priključena. Vsi razredi se kreirajo preko »factory« razredov, kar pomeni, da imamo na enem mestu v centralnem strežniku kreiranje razredov. To je predvsem pomembno pri podpori novih tipov krmilnih enot ali naprav.

4.3.4 Nivo predstavitvene logike

Implementirana je komunikacija med centralnim strežnikom in spletno aplikacijo. Uporabljamo Microsoft Remoting implementacijo, ki temelji na servisih in prenosu serializiranih razredov. Na centralnem strežniku imamo naslednje servise (Slika 4.11):

- Core servis je namenjen posredovanju podatkov o dinamični izgradnji spletne aplikacije,
- AuthenticationService servis je namenjen avtentikaciji uporabnika, omogoča več načinov prijave: z domenskim uporabnikom, uporabniškim imenom in z geslom ter certifikatom,
- AuthorizationService servis je namenjen avtorizaciji uporabnika, omogoča dva nivoja avtorizacije: nivo pred proženjem akcij, nivo pred dostopom do določene strani ali pogleda,
- UserService – servis je namenjen za delo z uporabniki pametnega sistema,
- EventServerService servis je predviden za delo z dogodki v pametni hiši,
- DeviceService servis je namenjen za delo z napravami: pregledovanje, urejanje in upravljanje. To je naš ključni servis, s katerim upravljamo in nadzorujemo celoten sistem.



Slika 4.11: Prikaz servisov, ki jih postavi centralni strežnik.

4.3.5 Predstavitveni nivo

Obravnavani nivo predstavlja grafični uporabniški vmesnik, s katerim lahko nadzorujemo in upravljamo sistem pametne hiše. Grafični uporabniški vmesnik se izvaja na napravi končnega uporabnika. To je lahko klasični računalnik, tablični računalnik, pametni telefon … V našem primeru bo predstavitveni nivo predstavljala spletna aplikacija, s katero bomo lahko sistem pametne hiše nadzorovali in upravljali.

4.4 Spletna aplikacija

Poleg izdelave centralnega strežnika in strežnika na krmilni enoti je cilj te diplomske naloge izdelati tudi spletno aplikacijo za nadzorovanje in krmiljenje naprav pametnega sistema. Zgrajena je s tehnologijo asp.net v povezavi s programskim jezikom C# verzije 4.0. Podatkovna baza teče na strežniku Microsoft SQL server 2012, do katere dostopamo preko centralnega strežnika. Pri izdelavi spletne aplikacije smo uporabili programsko orodje Microsoft Visual Studio 2012. V spletni aplikaciji smo uporabili tudi pakete »Jquery«, »Jquery-ui« …

4.4.1 Zgradba spletne aplikacije

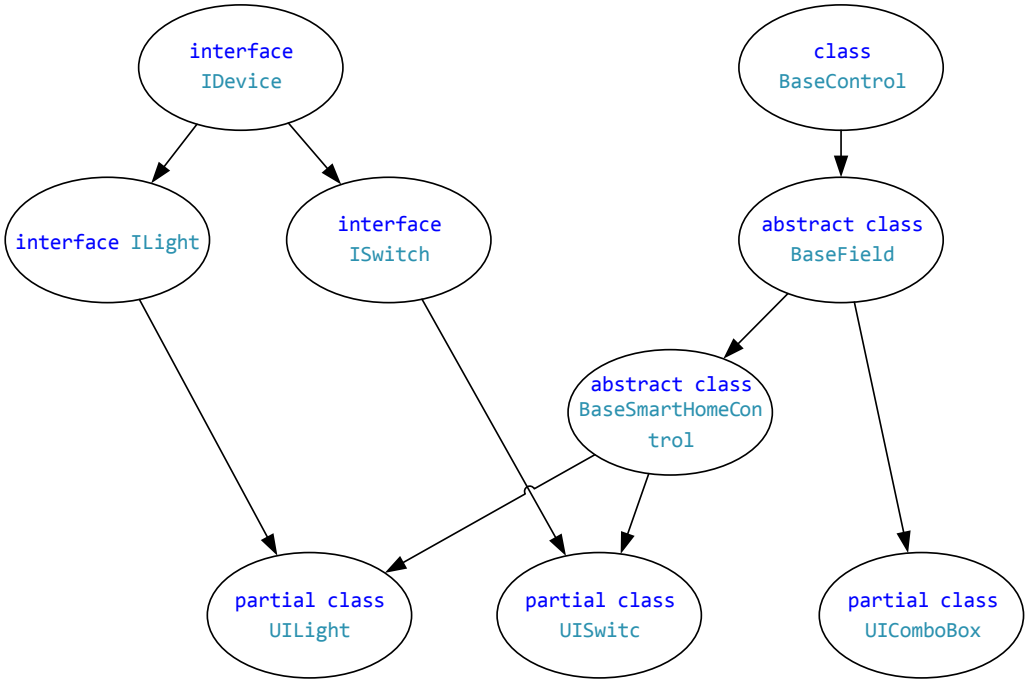
Centralni strežnik in spletna aplikacija sta zgrajena z n-tirno arhitekturo, kot prikazuje slika 4.7. Spletna aplikacija v n-tirni arhitekturi predstavlja predstavitveni nivo. Kot smo že omenili, aplikacija s strežnikom komunicira preko Microsoft Remoting protokola, ki v ozadju uporablja serializacijo razredov. Da bodo razredi serializabilni, jih moramo pravilno zgraditi. Pred razredom moramo podati »[Serializable]« (Slika 4.12). Vsi razredi, ki predstavljajo zapis v podatkovni bazi, so serializabilni, da jih lahko preko »Remoting« protokola pošiljamo spletni aplikaciji.

```
[Serializable]
 public class DeviceItem : BaseHomeItem
 {
     #region RECORD MEMBERS

     public long DeviceID { get; set; }
     public Guid DeviceGUID { get; set; }
     public long ControllerID { get; set; }
     public long DeviceTypeID { get; set; }
     public string Name { get; set; }
     public string Description { get; set; }

     #endregion

}
```

Slika 4.12: Primer razreda, ki je serializibilen.

Celotna zasnova spletne aplikacije je narejena z ASCX-kontrolami. V aplikaciji imamo nekaj splošnih ASCX-kontrol, kot so »CheckBox«, »ComboBox«, »Date«, »Label«…, kot tudi kontrole, ki predstavljajo naše naprave »UILight«, »UISwitch«, »UIBoard« … Vse kontrole dedujejo skupne razrede in tako se v aplikaciji obnašajo enotno. Dedovanje kontrol v sami aplikacije je prikazano na sliki 4.13:

- razred »BaseControl« vsebuje implementacijo dela s sejami uporabnika,
- abstraktni razred »BaseField« vsebuje implementacijo grafičnega izrisa elementa in implementacijo skupnih metod vsem kontrolam v spletni aplikaciji,
- abstraktni razred »baseSmartHomeControl« vsebuje splošne implementacije, namenjene vsem kontrolam, ki predstavljajo naše naprave,
- »UILight« in »UISwitch« sta naši ASCX-kontroli, ki vsebujeta svoje specifike in imata preko »interface« razreda predpisano, katere metode morata implementirati.
- »UIComboBox« je ASCX-kontrola, ki nam služi za izbiro sistema in kategorije.



Slika 4.13: Dedovanje »custom« kontrol.

Zakaj potrebujemo »interface« razrede, smo si ogledali že v poglavju 4.3.3, kjer smo predstavili zgradbo razredov v centralnem strežniku.

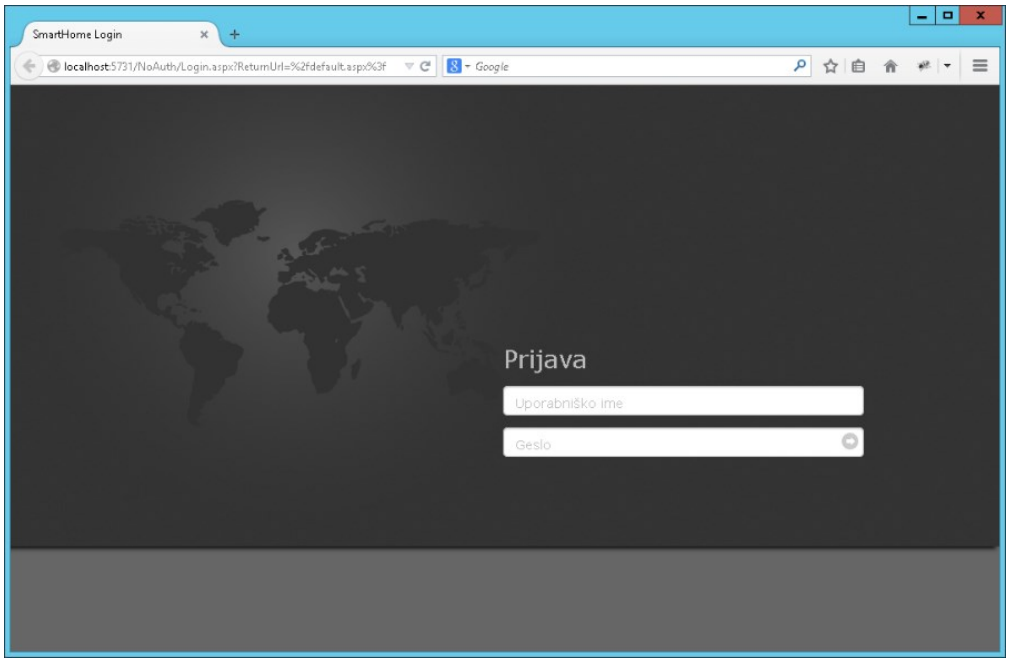
4.4.2 Predstavitev spletne aplikacije

Osnovni namen spletne aplikacije je uporabnikom pametnega sistema omogočati nadzorovanje in krmiljenje končnih naprav. Spletna aplikacija je zasnovana večuporabniško, kar pomeni, da omogoča delo z več različnimi tipi uporabnikov. To so administratorji, uporabniki, ki imajo možnost samo pregledovanja … Poleg dela s končnimi napravami lahko v spletni aplikaciji tudi urejamo uporabnike. V nadaljevanju bomo z uporabo spletne aplikacije prikazali osnovne funkcionalnosti, ki jih le-ta ima:

- Vstop v aplikacijo

  Avtentikacija (Slika 4.14) služi za pridobitev dostopa do sistema pametne hiše. Ta proces preveri identiteto uporabnika. Spletna aplikacija omogoča več načinov prijave:

  - možnost prijave z uporabniškim imenom in geslom,
  - možnost avtomatske prijave z domenskim uporabnikom,
  - možnost prijave z uporabo certifikata.



Slika 4.14: Vstop v spletno aplikacijo pametne hiše.

- Uporabniške vloge in njihovo urejanje

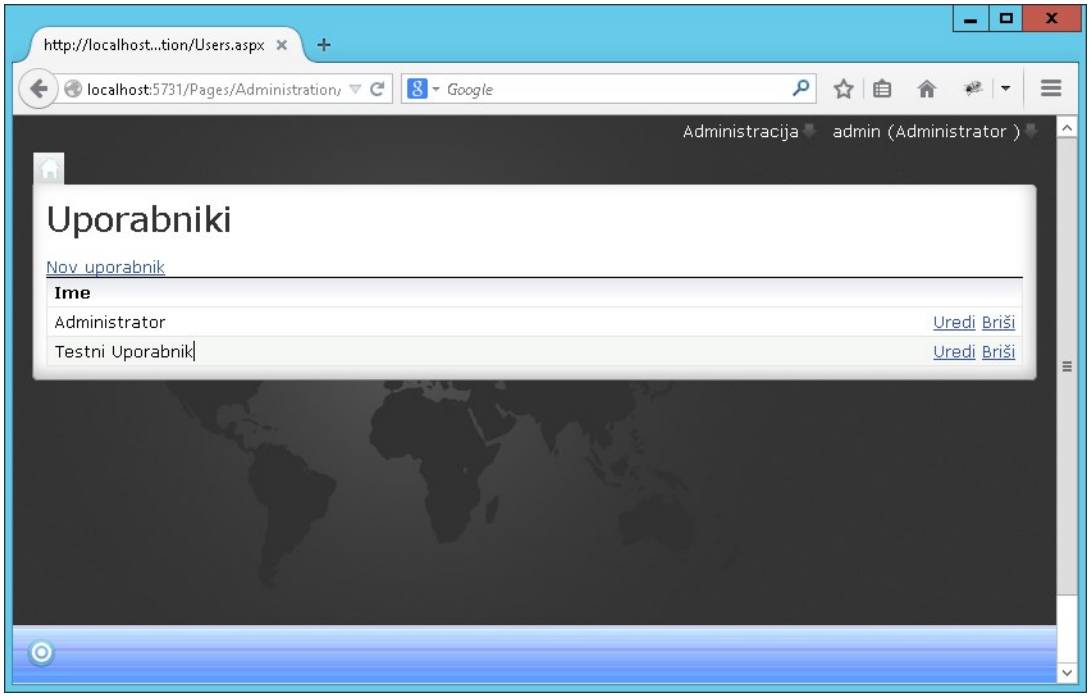
  Spletna aplikacija ima poleg avtentikacije implementirano tudi delo z uporabniškimi vlogami.

Uporabniške vloge definirajo:

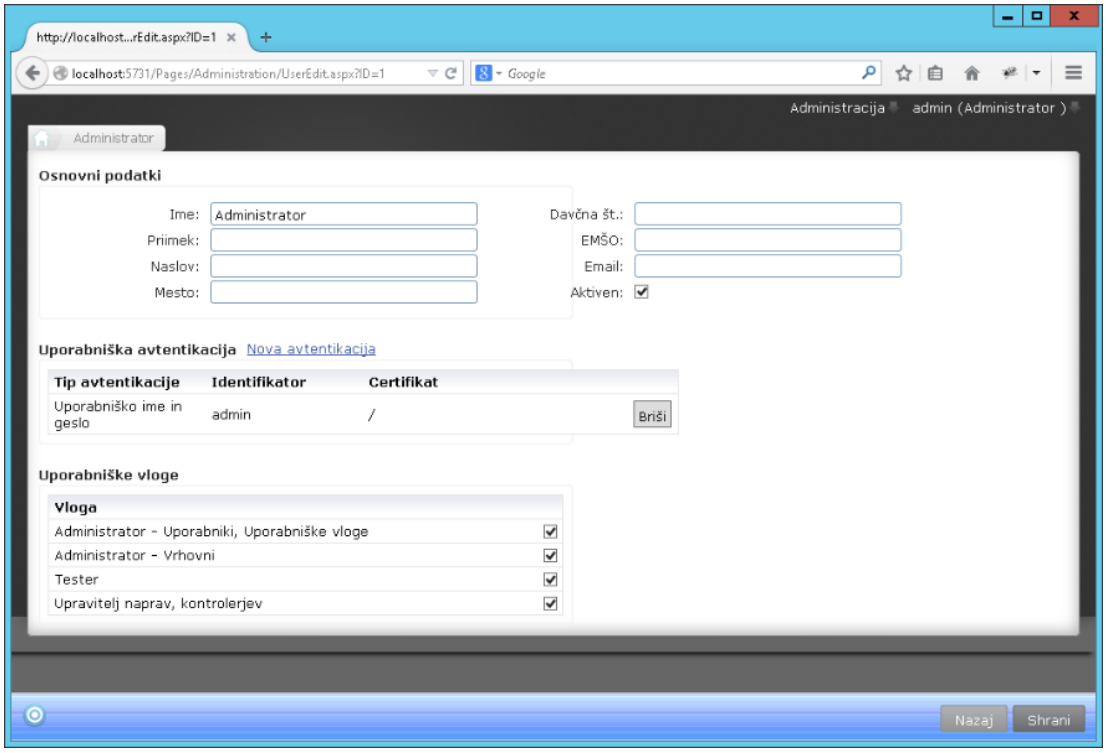
- akcije, nad katerimi ima uporabnik z izbrano vlogo pravico izvajanja,
- mape, strani in pogledov do katerih ima uporabnik dostop.

Primer: če vloga »Administrator - urejevalec naprav« definira dostop do administracijskega dela, kjer se ureja naprave, krmilne enote ... hkrati aktivira posamezne akcije, to pomeni, da uporabnik brez te vloge do tega dela administracije ne bo mogel dostopati. Poleg tega bo lahko izvedel le tiste akcije, ki jih vloga definira.

Vse uporabnike pametne hiše je možno pregledovati (Slika 4.15), urejati (Slika 4.16) in dodajati.



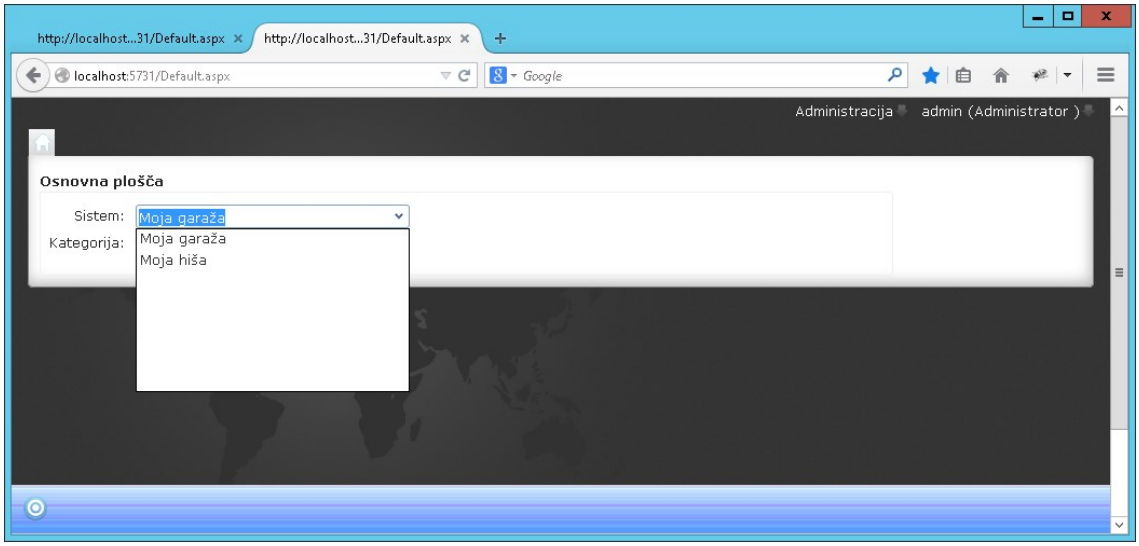
Slika 4.15: Pregled uporabnikov pametne hiše.



Slika 4.16: Urejanje uporabnikov pametne hiše.

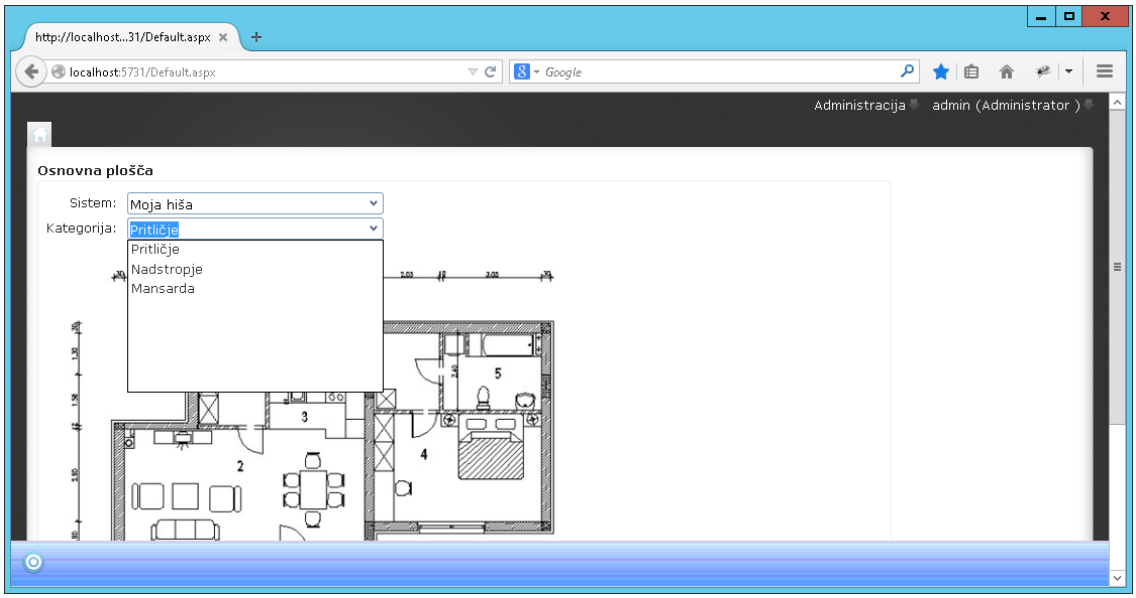
- Osnovna plošča

  Ko se prijavimo v spletno aplikacijo, se nam najprej odpre osnovna plošča, kjer imamo na izbiro sistem, s katerega želimo nadzorovati in upravljati. Znotraj vsakega sistema imamo tudi možnosti izbire kategorije. Celotna pametna hiša je zasnovana tako, da je možnih več sistemov (Slika 4.17) znotraj celotne hiše. Recimo, da imamo več objektov, ki jih želimo upravljati s pametno hišo: hišo in garažo. V sistem bi tako vnesli, da imamo za vsak objekt svoj sistem pametne hiše. Lahko imamo vse v enem, lahko pa ločeno. V tej aplikaciji z izbiro sistema določimo, kateri sistem želimo nadzorovati in upravljati.



Slika 4.17: Izbira sistemov v pametni hiši.

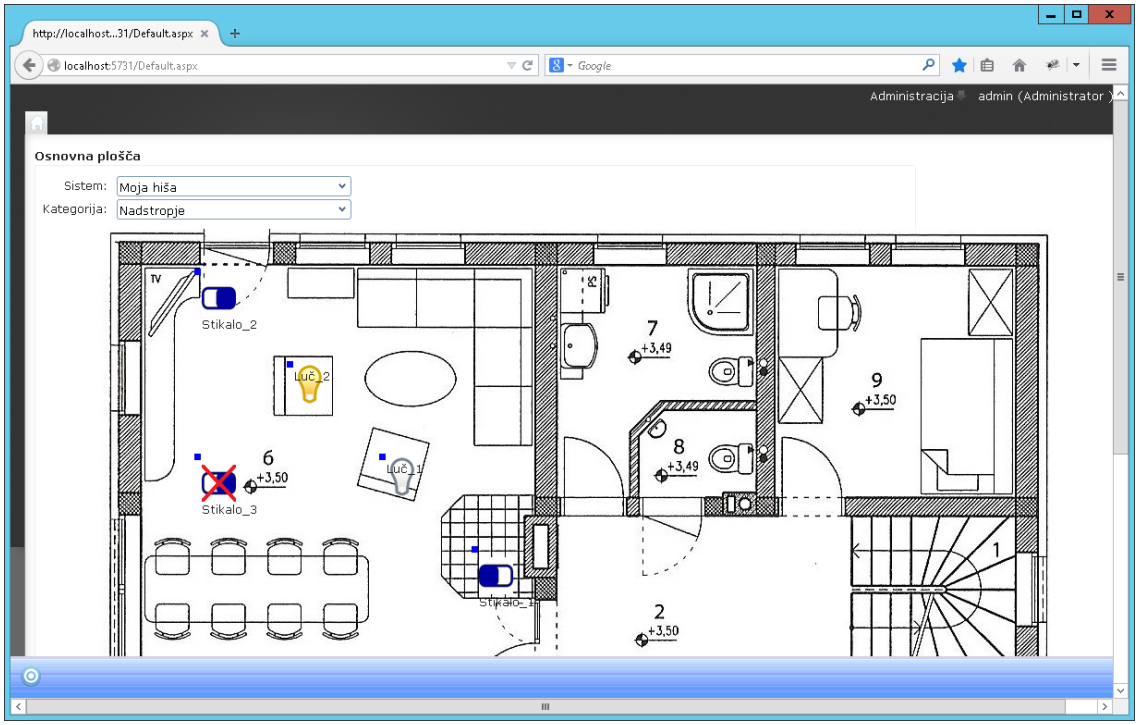
Znotraj vsakega sistema pa lahko imamo tudi več kategorij (Slika 4.18). Vsaka kategorija lahko predstavlja svoj del sistema: npr. pritličje, mansarda ali nadstropje. Lahko bi kategorije razdelili tudi po prostorih: kuhinja, kopalnica … Celotna pametna hiša je tako z vidika pregledovanja in krmiljenja razdeljena na več sklopov, kar nam nudi predvsem boljšo preglednost.



Slika 4.18: Izbira kategorije za izbrani sistem.

Slika 4.19 nam prikazuje, kako preko spletnega vmesnika pregledujemo in urejamo naprave, ki jih imamo registrirane v sistemu. Če v sistemu v določeni kategoriji nimamo nobene naprave, vidimo samo prazno osnovno ploščo, ki je v našem primeru tloris hiše.

V sistemu imamo dodana tri stikala in dve luči. Stikalo_1 prižiga Luč_1, Stikalo_2 pa Luč_2. Stikalo_3 je namenjeno prižiganju luči v mansardi, vendar krmilna enota trenutno ni dosegljiva, zato jo aplikacija označi kot nedostopno. To nam označuje rdeči križec, ki se je pojavil čez napravo.



Slika 4.19: Urejanje in pregledovanje končnih naprav pametne hiše.

Podrobneje si oglejmo Stikalo_1 in Luč_1. V sistem so vse naprave registrirane kot ločene. Kako naprave specificirati smo si pogledali v poglavju 3.3 Luč ima samo en fizični priključek, ki je priključen na en vhodno/izhodni priključek krmilne enote. Ta priključek ima definirano samo eno akcijo, ki jo označimo z GET_LIGHT_STATE. V podatkovnem modelu je za vsako napravo, ki je registrirana v sistemu, dodan svoj zapis v tabelo »homeUIElement« (Slika 3.9). Zapis v tej tabeli predstavlja grafični element na spletni aplikaciji. Če pa bi napravi Stikalo_1 in Luč_1 že v začetku specificirali kot eno napravo, bi jo na osnovni plošči videli kot eno napravo, ki bi predstavljala luč s stikalom.

- Podpora nove naprave

Poglejmo, kaj vse bi bilo potrebno razviti v spletni aplikaciji, če bi želeli podpreti nov tip naprave (recimo senčilo). Razvili bi samo ASCX-kontrolo na spletni aplikaciji, v centralnem strežniku pa bi implementirali nov razred »Sencilo«. Ta razred bi tako kot »Light« (Slika 4.10) podedoval vse ustrezne razrede, kot smo si jih podrobneje pogledali

pri centralnem strežniku. Po specifikacijah iz poglavja 3.3 bi napravo specificirali in vnesli v podatkovno bazo. Pri naslednjem vstopu v spletno aplikacijo bi tako napravo lahko upravljali kot naprave s slike 4.19.

4.5 Programski modul za krmilno enoto

Programska oprema na krmilni enoti je zelo pomembna, saj centralni strežnik preko strežnika na krmilni enoti pridobiva ali pa nastavlja podatke končnim napravam oziroma njenim vhodno/izhodnim priključkom.

Strežnik na krmilni enoti smo poimenovali »dcServer«. Da bi bil ta strežnik karseda neodvisen od krmilne enote, smo ga razvili v programskem jeziku Java, ki nam omogoča, da lahko isti strežnik teče na več različnih tipih krmilnih naprav. V nadaljevanju si bomo podrobneje pogledali, kako je zgrajen »dcServer« in kako mora biti zgrajena dinamična knjižnica, ki jo uporablja ta strežnik.

4.5.1 Zgradba strežnika

Za izgradnjo strežnika smo uporabili zunanjo knjižnico jsonrpc4j, ki nam ponuja enostavno implementacijo oddaljenih klicev JSON-RPC v Java programskem jeziku. Jsonrpc4j v ozadju uporablja »Jakson« knjižnico za pretvarjanje Java razredov v serializirano obliko in serializirano obliko v Java razrede. Te razrede mora v serializirani obliki pošiljati centralnemu strežniku, le-ta pa mora tudi v serializirani obliki pošiljati zahteve oziroma tako imenovane »Requeste«.

- Primer zahteve, ki jo pošlje centralni strežnik krmilni enoti:

  ```
  {"id":7795,"method":"RCGpioService.getValue","params":[1]}
  ```

  Centralni strežnik je poslal razred »Request« z naslednjimi spremenljivkami:

  - Id, katere vrednost je 7795 tipa »integer«,
  - Method, katere vrednost je " RCGpioService.getValue " tipa »string«,
  - Params, katere vrednost je razred »integer« z vrednostjo 1.

- Primer odgovora, ki ga pošlje krmilna enota centralnemu strežniku:

  ```
  {"jsonrpc":"2.0","id":7795,"result":"1"}
  ```

Krmilna enota je poslala razred »Response« z naslednjimi spremenljivkami:

- Jsonrpc, katere vrednost je "2.0" in je tipa »string«,
- Id, katere vrednost je 7795 tipa »integer«,
- Result, katere vrednost je "1" in je tipa »string«.

Jsonrpc4j knjižnica ima tudi enostavno implementacijo TCP/IP-strežnika s podporo več servisom, kar pomeni, da lahko temu strežniku implementiramo več servisov, ki delajo sočasno. V zgornjem primeru smo videli, da je centralni strežnik poslal zahtevo servisu »RCGpioService«, da izvede metodo »getValue«.

V nadaljevanju si bomo pogledali, kateri so ključni deli strežnika krmilne enote in kako so implementirani, da je strežnik čimbolj preprost (Slika 4.20):

```
//Ustvarimo objekte za komunikacijo
ServerSocket serverSocket =
ServerSocketFactory.getDefault().createServerSocket(Integer.parseInt(conf.getPort()), 0,
                                                     InetAddress.getByName(conf.getIP()));

//Ustvarimo objekte ki predstavljajo naše servise
IO io = initIOLibrary(conf.getIOLib(), conf.getExternalLibDirectory());
ITest test = new Test();
ISystemInfo systemInfo = new SystemInfo();

//Naše servis objekte dodelimo json-rpc strežniku
JsonRpcMultiServer jsonRpcMultiServer = new JsonRpcMultiServer();
jsonRpcMultiServer.addService("RCGpioService", io, IIO.class);
jsonRpcMultiServer.addService("testService", test, ITest.class);
jsonRpcMultiServer.addService("RCSystemService", systemInfo, ISystemInfo.class);

//Ustvarimo json-rpc strežnik in ga zaženemo
_streamServer = new StreamServer(jsonRpcMultiServer, 5, serverSocket);
_streamServer.start();
```

Slika 4.20: Programska koda strežnika krmilne enote.

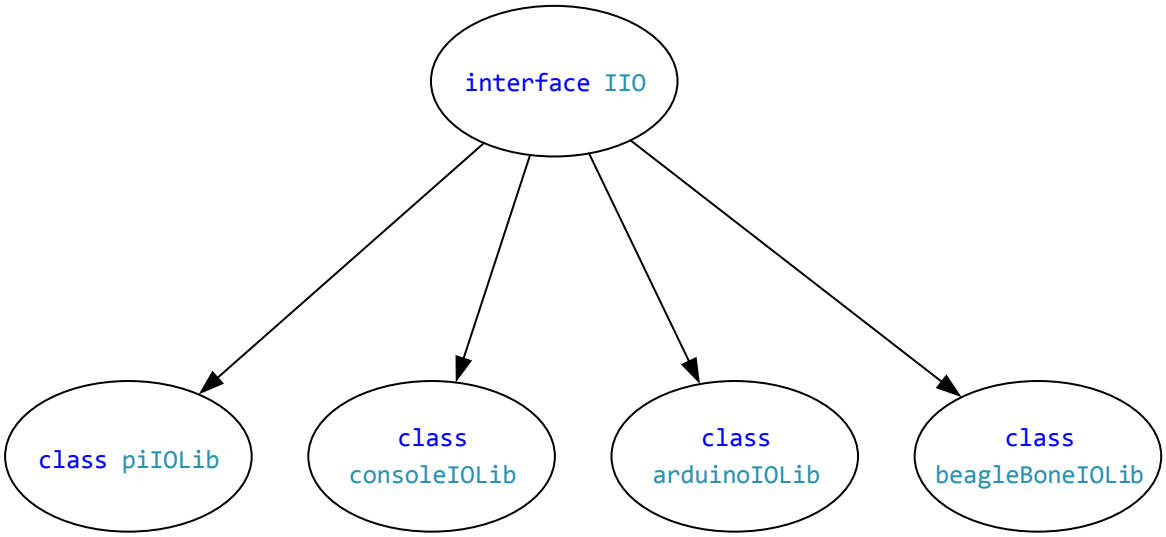
Pri implementaciji strežnika moramo najprej postaviti razrede za komunikacijo, ki poslušajo na določenem portu. TCP/IP naslov in port preberemo iz »conf« razreda, ki vsebuje podatke iz naše nastavitvene datoteke.

Nato moramo ustvariti razrede, ki predstavljajo naše servise. Vhodno/izhodni servis se naloži z metodo »initIOLibrary« iz dinamične vhodno/izhodne knjižnice. Te servise registriramo JSON-RPC-strežniku in zgradimo razred strežnika ter ga zaženemo (Slika 4.20).

4.5.2 Zgradba dinamične knjižnice

Naloga dinamične knjižnice je, da vsebuje vse specifike, kako krmiliti vhodno/izhodne priključke. Ta naloga je iz samega strežnika krmilne enote izvzeta zato, da lahko isti strežnik uporabimo na n-tipih krmilnih enot. Vhodno/izhodna dinamična knjižnica je odvisna od tipa, krmilne enote.

Katere metode mora implementirati vsaka dinamična knjižnica, nam določa »IIO interface«. Vsaka vhodno izhodna knjižnica (Slika 4.21) mora implementirati vse metode, ki jih »IIO interface« predpisuje.



Slika 4.21: Implementacija »IIO interface«-razreda skozi dinamične knjižnice.

»IIO interface« razred predpisuje implementacijo metod, ki jih prikazuje Slika 4.22.

```
public interface IIO {
	public void initLib();

	public void init(int pinAddress, String direction, String value);
	public void export(int pinAddress);
	public void unexport(int pinAddress);
	public void setDirection(int pinAddress, String direction);
	public String getDirection(int pinAddress);
	public void setValue(int pinAddress, String value);
	public String getValue(int pinAddress);
}
```

Slika 4.22: Metode, ki jih predpisuje »IIO interface« razred.

Pregled pomena metod:

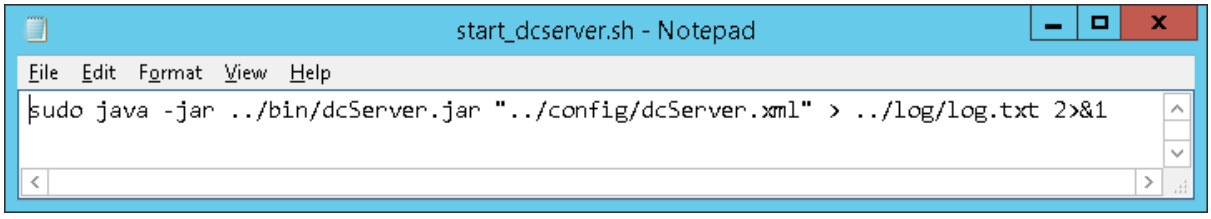
- »initLib« je metoda, ki jo kliče sam »dcServer« strežnik za postavitev inicialnih razredov,
- »init« metoda služi za inicializacijo vhodno/izhodnega priključka, vključno s smerjo in privzeto vrednostjo,
- »export« metoda služi za odprtje vhodno/izhodnega priključka,
- »unexport« metoda služi za zaprtje vhodno/izhodnega priključka,
- »setDirection« metoda služi za nastavljanje smeri vhodno/izhodnega priključka,
- »getDirection« metoda služi za branje nastavljene smeri vhodno/izhodnega priključka,
- »setValue« metoda služi za nastavljanje stanja vhodno/izhodnega priključka,
- »getValue« metoda služi za branje stanja vhodno/izhodnega priključka.

Vse metode razen »initLib« lahko z »RCGpioService« servisom kličemo preko JSON-RPC oddaljenih klicev.

Vsak razred s slike 4.21 vsebuje drugačno implementacijo metod. Skladno s tipom krmilne enote.

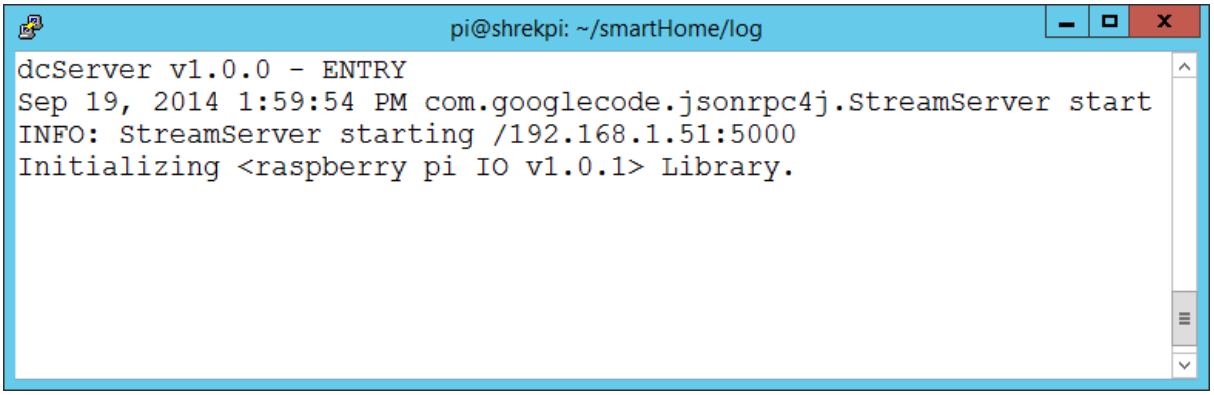
4.5.3 Zagon programskega modula krmilne enote

Zagonska skripta strežnika »dcServer« je »start_dcserver.sh«, katere vsebina je samo zagon jar-datoteke z nastavitveno datoteko, v kateri so podatki o TCP/IP-naslovu, portu in katero vhodno/izhodno knjižnico uporabljamo. Na krmilni enoti moramo zagnati vsak program z administratorskimi pravicami. Brez administratorskih pravic ne moremo dostopati do sistemskih vhodno/izhodnih datotek, zato imamo pred zagonom besedo »sudo«.



Slika 4.23: Vsebina zagonske skripte »dcServer« strežnika.

Za samodejni zagon strežnika, če krmilno enoto doleti izpad električne energije, moramo zagonsko skripto dodati v Cron. Cron [12] je časovnik, iz katerega operacijski sistem zaganja programe ali skripte. Operacijski sistem bo tako ob uspešnem zagonu avtomatsko zagnal skripto. Če je zagon uspešen, v log datoteko zapiše (Slika 4.24) verzijo, na katerem naslovu in portu posluša ter katero dinamično knjižnico je naložil in njeno verzijo.



Slika 4.24: Prikaz uspešnega zagona »dcServer« strežnika.

Poglavje 5 Sklepne ugotovitve

Cilj diplomske naloge je zasnovati in implementirati čimbolj modularen in dinamičen sistem pametne hiše, ki ga bomo lahko uporabili v svoji hiši. Da bi bil sistem, v katerega imamo lahko priključene poljubne naprave, čimbolj splošen in primeren za poljubno hišo, smo si pogledali, kakšne naprave so potrebne za njegovo delovanje.

Za centralno enoto je pomembna predvsem uporaba domačega strežnika, ki je namenjen poleg centralni enoti tega sistema tudi drugim funkcijam. Skozi zasnovo sistema smo spoznali, da bi bilo zelo težko vsako končno napravo v hiši povezati v eno točko sistema, od koder bi jih krmilili. Skozi iskanja rešitve težave smo spoznali kartične računalnike, ki smo jih uporabili za našo krmilno enoto. Le-ta pa se lahko nahaja na poljubni lokaciji v hiši in lahko jih imamo poljubno veliko. Za komunikacijo med krmilnimi enotami, centralnim strežnikom in napravami za nadzorovanje sistema pa smo uporabili kar domače omrežje.

Vsaka naprava, ki je del sistema, potrebuje tudi ustrezne programske module. Za programski modul krmilne enote nam je bilo predvsem pomembno, da je slednja preprosta in čimbolj neodvisna od tipa krmilne enote. Programski modul centralnega strežnika je veliko bolj kompleksen, saj mora biti sposoben procesiranja zahtev uporabnika, komunikacije s programskim modulom krmilne enote, uporabe podatkovne baze ter obdelovanja podatkov. Za nadzor in upravljanje naprav se nam je zdela najboljša rešitev uporaba spletnega brskalnika, saj tako postanemo neodvisni od naprave, s katero bomo upravljali in nadzorovali sistem.

Končne naprave so tako v sistem priključene na krmilne enote. Težavo nam je predstavljal opis: kako opisati nekaj, kar je priključeno na našo krmilno enoto, in razložiti, da nam le-to predstavlja končno napravo, ki jo bomo lahko upravljali ali nadzorovali. Tako smo skozi podrobno analizo ugotovili, da je potrebno predvsem dobro specificirati priključke naprave, njegove funkcije in uporabnost (za kaj ga lahko mi uporabimo). Ko smo spoznali, kako poljubno končno napravo priključimo, smo se lotili zasnove podatkovnega modela.

Podatkovni model mora poleg vseh informacij o končnih napravah, krmilnih enotah v sistemu vsebovati tudi informacije, kako je poljubna naprava prikazana na uporabniških vmesnikih, ki jih uporablja uporabnik sistema.

Pri implementaciji programskega modula krmilne enote smo brez oklevanja uporabili programski jezik Java, saj nam je ponujal ravno to, kar smo hoteli: enostavnost in neodvisnost od programske in strojne opreme. Program enkrat implementiramo in ga uporabimo na n-krmilnih enotah.

Pri implementaciji programskega modula centralne enote in spletne aplikacije smo imeli nekaj težav z njeno razvojno zasnovo in uporabo tehnologije. Predvsem smo se odločali, ali uporabiti programski jezik Java ali C# in kako implementirati strežnik ter aplikacijo. Ker pa pri centralnem strežniku ni toliko pomembna prenosljivost, smo se odločili za uporabo C# programskega jezika. Za implementacijo smo uporabili n-tirno arhitekturo, ki nam omogoča, da sta centralni strežnik in spletna aplikacija tudi z nivoja implementacije zgrajena kar se da modularno.

Rezultat dela je torej zasnova sistema pametne hiše, ki je čimbolj splošna, modularna in dinamično zastavljena.

Kot primer dobre prakse lahko navedemo, da smo v stanovanju tak sistem tudi fizično postavili in ga uporabljamo [6]. V sistemu imamo trenutno registrirani dve končni stikali in dve luči, ki jih lahko nadzorujemo in krmilimo. Skozi uporabo se je pokazalo, da je sistem zelo stabilen, pokazale pa so se tudi nekatere pomanjkljivosti predvsem v samem grafičnem vmesniku.

Seznam slik

Literatura

[1] Osvetlitev krajine. *ElekTrošt* [Online]. Dosegljivo: http://elektrost.si/osvetlitev_krajine.html.

[2] Samo čvrsta in vzdrževalna stopnišča ograj je varna. *Dnevnik* [Online]. Dosegljivo: http://www.dnevnik.si/dom/varnost/samo-cvrsta-in-vzdrzevana-stopniscna-ograja-je-varna.

[3] Senčila zaustavijo vročino in zmanjšajo stroške klime. *Dnevnik* [Online]. Dosegljivo: http://www.dnevnik.si/dom/energija/sencila-zaustavijo-vrocino-in-zmanjsajo-stroske-klime.

[4] How to use software PWM. *RasPi.TV* [Online]. Dosegljivo: http://raspi.tv/2013/rpi-gpio-0-5-2a-now-has-software-pwm-how-to-use-it.

[5] Kartični računalnik Raspberry Pi. *Raspberry Pi* [Online]. Dosegljivo: http://www.raspberrypi.org/.

[6] Predstavitev praktičnega dela diplomske naloge. Dosegljivo: https://www.dropbox.com/sh/2sgqswkpmjdndv6/AADqWPxsjPhvDXdWcU8_LRwIa?dl=0.

[7] Strateški načrt za optimizacijo poslovanja. BuyITC.si [Online]. Dosegljivo: http://www.buyitc.si/downloadfile.aspx?fileid=791.

[8] Microsoft Visual Studio. *Wikipedia* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Microsoft_Visual_Studio.

[9] Eclipse. *Wikipedia* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Eclipse_(software).

[10] Notepad++. *Wikipedia* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Notepad%2B%2B.

[11] Jar tip datoteke. *Wikipedia* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/JAR_(file_format).

[12] Časovnik za izvedbo nalog. *Wikipedia* [Online]. Dosegljivo: http://en.wikipedia.org/wiki/Cron